AUTOMATED DRIVER DROWSINESS DETECTION SYSTEM

Enrol No. (s) – 9919103206, 9919103219, 9919103222
Name of Student (s) – Prakhar Jauhari, Shikhar Gupta, Manav Choudhary
Name of supervisors(s) – Dr. Himanshu Agrawal

MAY- 2023

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

in

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

TABLE OF CONTENTS

DECLARATION

We hereby declare that this submission is our original work and that, to the best of our knowledge and belief, it does not contain any earlier publications or writings by other people, nor does it contain any writings that have been approved for the award of any other degree or diploma by the university or other institution of higher learning, unless appropriate attribution has been made in the text.

Place: Noida	Prakhar Jauhari	(9919103206)
Date: 10 May 2023	Manav Choudhary	(9919103222)
	Shikhar Gupta	(9919103219)

CERTIFICATE

This is to attest that I oversaw the work that was completed for the "Automated Driver Drowsiness Detection System" project that Prakhar Jauhari, Manav Choudhary, and Shikhar Gupta submitted in partial fulfilment for the award of a degree in B.Tech from Jaypee Institute of Information Technology, Noida. For the purpose of awarding this or any other degree or diploma, this work has not been submitted in whole or in part to another university or institute.

Signature of Supervisor:

Name of Supervisor	: Dr. Himanshu Agrawal
Designation	: Assistant Professor
Date	: 10 May 2023

ACKNOWLEDGEMENT

We would first and foremost like to express our gratitude to our mentor, Dr. Himanshu Agrawal, Assistant Professor at the Jaypee Institute of Information Technology in Noida, for his thoughtful and effective guidance throughout this project. He gave us the freedom to work at our own pace and on our own terms while also offering us incredibly helpful directions when they were called for. We also want to express our gratitude to our friends and classmates who served as excellent sources of inspiration and support for the duration of this endeavour. We extend our deepest gratitude to everyone else who assisted us with this initiative, whether they were aware of it or not.

Signatures of Students

Prakhar Jauhari	(9919103206)
Manav Choudhary	(9919103222)
Shikhar Gupta	(9919103219)

SUMMARY

Every year many people lose their lives due to fatal road accidents around the world and drowsy driving is one of the primary causes of road accidents and death. Fatigue and micro sleep at the driving controls are often the root cause of serious accidents. Most of the traditional methods to detect drowsiness are based on behavioral aspects while some are intrusive and may distract drivers, while some require expensive sensors. Therefore, in this project, a light-weight driver's drowsiness detection system is developed and implemented.

The system records the videos and detects driver's face in every frame by employing image processing techniques. The system is capable of detecting facial landmarks, computes Eye Aspect Ratio (EAR) and Eye Closure Ratio (ECR) to detect driver's drowsiness based on adaptive threshold. The feature point detector use 68-point facial landmark, but we constrain landmarks to the convex hull. Results show high sensitivity of the algorithm in the tests performed on a vehicle with a webcam and a warning alert, and produced good results with more sensitive conditions.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Sr. No	TITLE	PAGE NO.
1	LBP – Local Binary Pattern	1
2	ECG – Electrocardiogram	2
3	EMG – Electromyogram	2
4	ANN- Artificial Neural Network	2
5	EAR- Eye Aspect Ratio	3
7	CGM - Constrained Generative Model	6

1. INTRODUCTION

1.1 General Introduction

Driver Drowsiness Detection Technique is a car safety automation system to mitigate the number of accidents when a driver is found lethargic or sleepy. Driver fatigue and drowsiness can lead to severe physical lifetime injuries and even prove out to be fatal**.** Up to twenty percent of deadly crashes can be connected to drowsiness while driving. Because of the various threats posed due to drowsiness, counteracting means need to be developed and adopted to reduce the hazardous side-effects. If a driver is not alert, it can be due to over exhaustion or intoxication. The initiation of automation for deterring drowsiness is a first-rate project inside the discipline of accident evasion structures. An endeavor to design a system has been made toward the driver's face to be able to identify fatigue. After the detection of fatigue on the face of the driver a caution sign is issued to alert the driver and a penalty is imposed on the driver for the fatal misconduct. A system which includes a video camera is fitted in the driver's vehicle, which captures the live video and continuously converts it into frames to send it to determine whether a driver is sleepy or not. A condition for eye aspect ratio values are considered in determining the eye state i.e. open or close. The set of rules is proposed, implemented, tested, and discovered operating satisfactorily.

The automated drowsiness detection system has been advanced with the usage of camera that points immediately closer to the driving force's face and monitors the driving force's eyes. The LBP (Local Binary Pattern) algorithm detects the eyes, and additionally the way to determine if the eyes are open or closed.

Countermeasure tools are presently required in sleepiness associated accident prevention. The system performs real-time processing on the frames that are sent by the video camera while live streaming to evaluate whether a driver is lethargic on not. Once the conditions required to determine the drowsiness are met, the system then generates an alarm on the hardware connected in the system to alert the driver and wake them up at once. There is large number of techniques and variations present to determine the drowsiness of a driver. Some of these techniques include Image processing to detect facial features, Electroencephalography which monitors our body activities, Neural Network based detection [7], vehicle monitoring detection. [4], physiology based, behavior based etc.

In this age where human being is breaking new barriers in development of technology every day, protection and safety of life and property has arisen as a great avenue for advancement of technology. Automobiles are one such technological platform which require constant advancement in order to become safer for their users and less prone to theft and misuse.

1.2 Problem Statement

When traveling long distances, there is a risk that the driver may become inattentive due to drowsiness, which could result in serious accidents. Therefore, we need a system for detecting drowsiness and avoid these accidents. We intend the proposal to implement an Automated Drowsiness Detection System for protection of user and secure vehicle access which will generate a penalty over the driver found drowsy during working hours and preventing the intruders to access the vehicle by appropriate authentication.

1.3 Significance of Problem

On lengthy trips, it is possible for the driver to become distracted by drowsiness, which could potentially result in fatal collisions. Systems like automatic Driver Drowsiness Detection make it feasible to identify a driver's bodily behaviours while operating a vehicle that could be dangerous to the occupants or those nearby. The suggested solution will improve the technological position of the current cab service providers while reducing the frequency of traffic incidents.

1.4 Empirical Study

Before starting to work on the project driver drowsiness detection system, we studied various techniques and algorithms used so far to detect drowsiness. We detected the drivers face from the video stream and then detected eye using eye aspect ratio. Techniques which we studied includes Local Binary Pattern (LBP), Viola Jones Algorithm, computed heart rate variations using logistics regression in machine learning, some of the behavioral based techniques including head movements, yawning were also considered to detect driver drowsiness. A new method which includes detection of pulse rates, heart rates with the help of smart watch and calculating ECG, EMG values to compare it with the threshold value. Traffic signals were also used and applied ANN to classify them. The angles of steering wheel and how the movement of vehicle is occurring observed.

1.5 Brief description of the Solution Approach

After procuring the live streaming of the driver’s face, the system will convert it into frames for further processing. The Local Binary Pattern (LBP) algorithm will be applied for the detection of the face portion in the image i.e. the eyes Drowsiness will be determined from whether the eyes of the driver are closed for an instance of seconds. The device will be able to distinguish between a motorist who is drowsy and a driver who is not drowsy by analysing the eye states. The driver's facial position and the condition of his or her ocular features will next be classified by classifiers. An alarm will sound if a drowsy driver is found until the system determines the motorist is awake.

A python script is run through the kernel which also contains the eye detection trained model which will help in recognizing whether the driver is drowsy or not.

1.6 Comparison of existing approaches to the problem framed

As driver drowsiness is becoming one of the major causes of road accidents, numerous approaches designed so far in order to give alertness to the driver to get rid of the accidents. Techniques used like Viola Jones Algorithm which process the image and return the location of features with high accuracy. In the same manner we used LBP (Local Binary Pattern) to detect the facial structures.

Other than that, there are various approaches including vehicle based approach in which deviations of lane, sudden movement of the steering wheel, pressure on the acceleration pedal are some of the measures to be considered and if the steering wheel characteristic value crosses specified threshold the driver will be alarmed. Behavior based approach is also used for detection includes eye closure, yawning, eye blinking, head pose which we are considering in our project. Among the different behavioral aspects, we considered eye aspect ratio (EAR) through which will calculate the value and compare it with the provided threshold. We used 68-point classifier to mark the points. If the driver found to be drowsy alarm will be activated.

Other approaches which can also be used are physiological based which leads to deal with the physiological signals like electrocardiogram (ECG) and electromyogram (EMG). The reliability and accuracy of this method is very high but requires sensors in order to study these attributes leads to complexity.

2. LITERATURE SURVEY

2.1 Summary of the papers studied

At the moment, drowsy driving is one of the main causes of traffic accidents. According to statistics, drowsy driving causes a significant portion of traffic accidents, many of which end in fatalities and serious injuries. The creation of these systems that can assess driver weariness and alert him in advance has been influenced by numerous studies on design formulation in an effort to solve the following. This prevents the driver from nodding off behind the wheel and causing an accident. Some conventional methods used measurements based on the vehicle to design their systems, but these measurements are greatly influenced by the design of the road, the type of vehicle, and the driving ability. Additional techniques that tend to offer better accuracy in tracking the driver's drowsiness used psychological measures for their system. However, since electrodes must be applied to the head and body, such methods are typically invasive. Additionally, there aren't many studies out there that use subjective measurements as the system's input, but doing so can distract drivers and produce unclear results. We have all become extremely networked because to the Internet of Things and other technological advancements. The movement of the driver can be tracked by sensors like alcohol sensors and drowsiness sensors on the steering wheel, which can also control an alert to wake the driver up.

Various approaches have been classified into various categories mentioned below:

i. **Vehicle Based** – This method continuously monitors characteristics like as sudden steering wheel movements, pressure on the accelerator, and lane deviations to see if one or more are crossing a certain threshold, which indicates that the driver is drowsy. However, studies have demonstrated that these indicators are subpar drowsiness predictors. Additionally, it might be too late to stop an accident if the driver begins to stray from the lane.

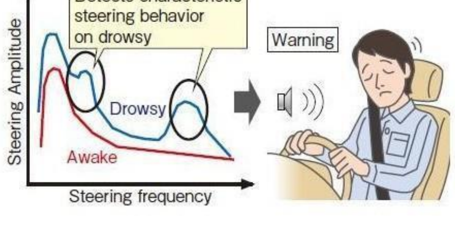


Figure 1. Steering wheel Based detection [paper 4]

ii. **Behavior Based-** A camera is utilized to monitor the driver's behavior, such as yawning, eye closure, eye blinking, and head pose, among others. If any of these signs of drowsiness are detected, the system alerts the driver. However, this vision-based approach has limitations. One of the main constraints is lighting since regular cameras do not function well in low light conditions, such as at night.

iii. **Physiological Based-** The driver is informed if any of the symptoms are noticed when the behaviour, such as groaning, ocular closure by eye blinking, head position, etc., is being watched. The relationship between physiological signs, such as electromyograms and electrocardiograms and and driver drowsiness is investigated. This method has a very high level of accuracy and reliability when compared to other methods**.**

iv.

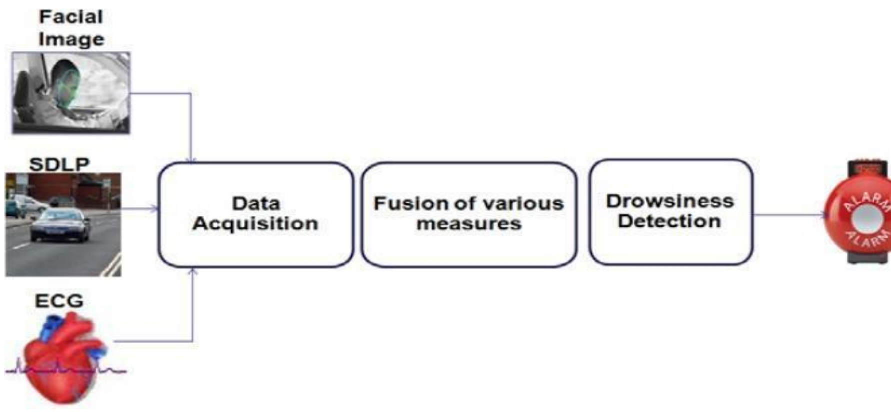


Figure 2. Various Drowsiness Detection Techniques [paper 28]

iv. **Yawning Based Technique-** It calls for a number of processes, including real-time recognition and tracking of the driver's confront, recognition and monitoring of the mouth contour, and detection of yawning displays based on calculating the pace and magnitude of changes in the mouth contour region. Connive Corp. created the APEX platform for vehicle smart cameras. In our method, a surveillance camera that is mounted inside the car according to the front mirror constantly captures the driver's face.

v.

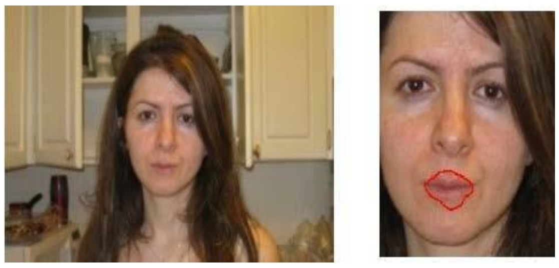


Figure 3. Yawning Based Technique [paper 14]

2.2 Integrated Summary of the Literature studied

The objective of the paper by Cn.K et al. [1] is to use the Viola-Jones algorithm to detect the presence of a face in a given image accurately. Face detection has been extensively studied in the field of computer vision, and while it is an easy task for humans, it poses a significant challenge for computers due to various factors such as variations in scale, location, viewpoint, illumination, and occlusions. Although numerous methods have been developed for face detection, the Viola and Jones algorithm is considered to have the most significant impact in recent years. This algorithm is capable of processing images quickly and achieving high detection rates, making it a popular choice for face detection applications.

The main objective of the project was to understand and implement the Viola-Jones algorithm and conduct experiments to enhance its performance. In another study by Féraud, R., Bernier, O.J et al. [2], the challenge of detecting faces in complex images was addressed using a neural network model called the Constrained Generative Model (CGM). This model can even detect side faces accurately. To improve the quality of the model, counterexamples were used, and a conditional mixture of networks was employed to reduce false alarms and detect side view faces. Furthermore, a fast search algorithm was proposed to decrease the computational time cost. Huang, D. et al. [3] focused on classifying demographics, including face detection, expression recognition, and analysis using Local Binary Pattern (LBP) and similar techniques.

Krajewski, J., Sommer, D., et al. [4] provide an alternative approach. This study illustrates a sleepiness detection system based on steering-wheel behaviour that looks at constant lane drifting while driving and fast reactions. This study looked at a fatigue surveillance system based on steering behaviour. The benefit of employing steering behaviour for fatigue detection is that these devices determine continuously, affordably, unobtrusively, and robustly even under the most challenging environmental conditions. They computed three feature sets in the time, frequency, and state space domain (a total of 1251 features) using advanced methods of signal processing for feature extraction to detect fatigue-impaired steering patterns..

Five machine learning approaches (such as Support Vector Machine and K-Nearest Neighbour) were used to process each feature set separately. An overall classification value was created by combining the results of each individual classifier. In the end, they created a meta-ensemble class value by combining the combined values of three feature subsets. Driving samples from a driving simulator are collected during a sleep deprivation study (N=12) in order to validate the steering behaviour analysis. They successfully differentiated between light and heavy fatigue with an accuracy rating of 86.1%. Action recognition and tracking are typically treated as two separate problems in typical template-based action recognition systems, and they are each solved individually. The method, however, emphasises how tracking and action recognition may be closely combined into a unified framework, where tracking helps with action recognition and vice versa.
In order to gauge a driver's level of attentiveness, A.M. et al. [5] watch video-based inputs and look for facial expressions as well as the ratio of open to closed eyes.

The assessment approach is provided by Saini, V. et al. [6] and analyses the driver's sleepiness based on a number of factors, including speed for a predetermined amount of time and the most recent breaks taken, and it provides the driver a drowsiness alarm. Long-distance drivers who don't stop frequently run a greater danger of becoming sleepy. According to experts, this is a state that they usually fail to recognise at an early enough stage. According to studies, drowsy driving causes more accidents on the road than drunk driving, accounting for about one-fourth of all serious motorway accidents. In addition to alerting drivers to their current level of weariness, attention assist can detect drowsiness and inattentiveness over a wide speed range. and the distance travelled since the previous rest stop. In order to create a more awake and aware driver, Soukupova, T. et al. [7] suggested a method to detect facial features to estimate the ratio of fully-opened, partially-opened, and closed eyes as well as to assess the rate at which the driver blinks.

A real-time process is proposed for identifying the blinking of an eye in a video clip shot with an average camera. Recent landmark detectors show excellent sensitivity despite head orientation in relation to a camera, variable light, and facial expressions after being taught on in-the-wild datasets. They demonstrate that the landmarks may be identified with sufficient precision to determine the level of the eye movement. As a result, the suggested algorithm calculates the landmark locations and derives a single scalar number, the eye aspect ratio (EAR), which describes the eye opening in each frame. In a small temporal window, an SVM classifier may also identify eye blinks as an arrangement of EAR values.

The simple algorithm outperforms the state-of-the-art results on two standard datasets. S. Mehta *et al* [8] majorly focuses on developing a light weight, real-time driver drowsiness detection system and further implementing it on Android application. In the designed system, they record the video at the very first in order to detect driver's face in all the frames using image processing techniques which was sent to the local server. Using random forest classifier with an accuracy of 84%, they concluded to be the most efficient classifier for the above proposed approach.

P. Bharadwaj *et.al* [9] discussed mainly about implementation and working of the 'Driver drowsiness detection system using video audio alarm warning' focusing majorly on the cab drivers but have many other applications too. The system consists of a camera for face detection and calculating the EAR ratio. In the first step will detect fatigue, if detected the alarm will alert the driver. Alarm was connected with the GPIO port of raspberry pi. The complete process works in three major steps. In 1 st stage, driver's face has been detected and alarm was called. In 2 nd stage, they extracted the important facial features. In the 3 rd and last stage eye state has been monitored and checked that weather the eyes are open or close. R. P. Kumar *et. al* [10] Computer vision plays a vital role in driver drowsiness detection system. Different types of models were proposed to classify scale invariant feature (SIFT) and Dominant rotated local binary pattern (DRLBP). In this paper, they have used Artificial Neural Networks (ANN) to classify traffic signs.
At first they have extracted image of traffic signal and using SIFT extracted the facial features also. Further, they applied Back Propagation Network (BPN). SIFT and DRLBP are used as a combination in the proposed approach rather than using separately. Further they have monitored the eye state which tells about the state of fatigue as early as possible to avoid the accident**.**

While drowsiness detection the most important aspect to detect fatigue is the facial state like frequency of yawning, blinking of eyes are few of the gestures that can be measured. W. Deng *et. al* [11] designed a system called as Dri Care, which will detect the state of fatigue of the driver based on the blinking, yawning, span of eye blinking measures with the help of motion pictures (video images). So, they created a new tracking algorithm via overcoming the shortcomings of the previous one to improve the accuracy. B.-L. Lee *et. al* [12] designed a completely stand alone, distraction-free, and wearable system for driver drowsiness detection by incorporating the system in a smart watch. The main objective is to detect the driver's drowsiness level based on the driver behavior derived from the motion data collected from the built-in motion sensors in the smart watch, such as the accelerometer and the gyroscope.

Because different SVM models are used for different hands, this user-predefined system can be used by both left- and right-handed users. The method employed for detecting driver fatigue is reliable, secure, and distraction-free. A. Mittalet. al. [13] provides an overview of the methods to recognise driver tiredness through observation of driving behaviour. It is hypothesised that irregular driving patterns are caused by the driver being sleepy. For this, a variety of evaluations, including subjective, behavioural, physiological, and vehicular measures, have been used. The comparative examination of these methods reveals that behavioural interventions are simple to obtain and unobtrusive, so they don't bother the driver. The most accurate and useful behavioural measure, according to research, is the head movement measure. A method by T. Wang et al. [14] leveraging video analysis to determine driver weariness or drowsiness is given. The method of extracting driver yawning is the main topic of this study. The driver's face region is located using a real-time face detector. The Kalman filter is utilised following that to track the facial region. Additionally, to identify driver yawning in video, the mouth window is localised within the region of the face, and the degree of mouth flexibility is extracted based on mouth features. When occlusion or miss-detection occur, the system will reset.

M. Awais et al. [17] employed a hybrid approach to detect driver drowsiness by integrating physiological signals to enhance system performance and user comfort. They proposed a method to detect drowsiness in drivers by combining features extracted from electrocardiography (ECG) and electroencephalography (EEG) signals. The study involved measuring differences between alert and drowsy states using physiological data collected from 22 healthy subjects in a driving simulator-based study. To induce drowsiness in the participants, a monotonous driving environment was used. The features extracted from the ECG signal included heart rate (HR) and heart rate variability (HRV), including low frequency (LF), high frequency (HF), and LF/HF ratio. The results demonstrate that the proposed method can be an effective solution for a practical driver drowsiness system that is both accurate and comfortable to wear.

In addition to the methods discussed previously, there are other approaches that can be utilized for detection. For instance, M. Chai et al. [18] used a driving simulator to collect twelve parameters related to the steering wheel, selecting five parameters that had significant correlations with driver status using variance analysis. Based on the selection of these parameters, they built a multilevel ordered logit (MOL) model, support vector machine (SVM) model, and BP neural network (BP) model. Under the same classification conditions, the MOL model exhibited much higher recognition accuracy than the other two models, leading to the conclusion that the MOL model that considers differences among individuals and uses the steering wheel parameters outperforms other models in terms of driver state recognition. B.-H. Jang et al. [19] conducted a preliminary study for detecting drowsiness using posture and image processing technology, utilizing pressure sensors to study posture and investigating the possibility of drowsy recognition using histograms. The experiment confirmed that positions can be distinguished through pressure sensors, and the drowsiness phenomenon can be recognized using histograms. In contrast to conventional drowsiness detection methods, which are based on eye states alone, M. A. Assari et al. [20] used facial expressions to detect drowsiness.

Detecting drowsiness poses numerous challenges, such as variations in lighting conditions, and the presence of glasses or facial hair, among others. In order to address these challenges, a hardware system based on infrared light was proposed and implemented in a project. The system follows a face detection step, and then extracts and tracks facial components that are deemed the most significant and effective for detecting drowsiness in video sequence frames. The method was tested and deployed in real-world settings. Another non-intrusive method for detecting drowsiness is presented in a paper by Wei Zhang et al. [21]. This method utilizes eye-tracking and image processing and introduces a robust eye detection algorithm to tackle problems caused by changes in illumination and driver posture. Six measures are calculated from the eye data, including percentage of eyelid closure, maximum closure duration, blink frequency, average opening level of the eyes, opening velocity of the eyes, and closing velocity of the eyes.

On a real road, a 15-hour experiment is conducted, and the data gathered are segmented and labelled with three levels of fatigue: "awake," "drowsy," and "very drowsy." The experiment yielded an average fatigue detection accuracy of 88.02%, supporting the usefulness of the suggested approach for engineering applications. In this paper, Zhitao Xiao et al. [23] propose an eye state recognition method based on convolution neural network (CNN), calculate percentage of eyelid closure over the pupil over time (PERCLOS), and blink frequency to detect fatigue in order to solve the aforementioned problems and create the algorithm to maintain accuracy and real-time at the same time. Pressure measurements can also help in detecting fatigue, and a paper Saad Ahmed Sadi*et. al*[24]designed a system to detect the drowsiness through face monitoring techniques and pressure measurement.

Using the combination of numerous optimised indications based on driver physical and driving performance parameters, derived from ADAS (Advanced Driver Assistant Systems) in simulated settings, J. Javier Yebeset.al [26] provides a non-intrusive method for detecting driver sleepiness. Instead of long-term sleep/awake regulation prediction technology, the research focuses on real-time sleepiness detection technology. To provide reliable and optimised driving indications that may be utilised in simulators and future actual scenarios, they have created their own vision system. These signs are mostly dependent on the physical capabilities and driving abilities of the driver. A deep architecture called the deep drowsiness detection (DDD) network is suggested by S. Parket al. [27] for learning useful characteristics and sleepiness detection given an RGB input video of a driver. The Deep Deep Learning (DDD) network is made up of three deep networks that work together to provide global resilience against background and environmental fluctuations and to learn regional head and facial motions that are crucial for accurate recognition. This work by Wan-Young Chunget. al[28] aims to compare the classification performance of this technique with the traditional method, which employs fast Fourier transform (FFT)-based features, in order to categorise alert and sleepy driving episodes using the wavelet transform of HRV signals across short time periods.

The low-cost and user-friendly hardware platform proposed by L.-B. Chenet. al [29] has been designed to incorporate a driver drowsiness detection application in Ellcie-Healthy smart glasses. The wearable device's constrained battery capacity makes it challenging to strike a balance between performance and autonomy. In this study, the authors proposed an analytical power consumption model-based system-level modelling approach to estimate the device's autonomy while ensuring the accuracy of sensor measurements. The experiment's findings show how the suggested system-level power modelling approach can be used to strike the ideal balance between QoS and autonomy.

In this paper by Yong-Guk Kim et al. [30], they proposed an approach that uses a 3D convolutional neural network to extract features in the spatial-temporal domain, followed by gradient boosting for the classification of drowsiness, and finally semi-supervised learning to improve overall performance. The findings suggested that this approach has real-world applications.

3. Requirement Analysis and Solution Approach

3.1 Overall description of the project

The proposed system of automated driver drowsiness detection system is a non-intrusive automation forum to detect fatigue in driver of the vehicle by localizing the eye states and computing the eye aspect ratio and subsequently the alarm will be activated in case of ratio found less than the demographically set threshold. Further, driver authentication is also facilitated through efficient face detection algorithm Voila Jones, which will detect the facial features of the person driving the car and match it with the image of the driver included in the database. In case of an unmatched identity it will debar the access of the vehicle and set the alarm. This gives the security of the vehicle an added advantage. The system has been implemented on the Rasberry pi module for its hardware execution.

3.2 Requirement Analysis

Functional Requirements

i. **Software Requirement**

Python 3.6

Anaconda

Operating System: Linux

ii. **Hardware Requirement**

Processor: Intel i3 Processor or higher

RAM: 4GB or more

Camera

Following requirements should be met-

- To authenticate the driver from the database
- To determine the drowsiness of the person driving.
- Issue alert when drowsiness is detected
- Raise alarm in case of drowsiness detection.
- Successful delivery of SMS including the location of the driver to avoid accidents.

Non-Functional requirements

This project has following non-functional requirements:

• Reliability: The system must work with great precision and consistency at all times.
• Usability: It should be simple to implement the system configuration. Codes created for the system's execution should contain comments to make them easier to read for future development.
• Performance: The system has to work correctly in every circumstance. The system must have an inadequate runtime that is functional. Images taken by the smartphone camera should be clear and precise.

Logical database requirements

The project is based on real time data streaming hence no specific database requirement is necessitated. In addition, the database for the driver authentication can be made manually on the system and furthermore advancement of database systems can also be proposed for the future.

3.3 Solution Approach

This module aims at processing the acquired video images. The eyes are first recognised, and then the driver's face is located in the video stream. Then, using the pixel intensity difference and a threshold value, the eye's condition will be determined. Six coordinates of the eye are taken into account by The EAR algorithm (Eye Aspect Ratio). The denominator half of the EAR algorithm's formula corresponds to horizontal eye markings, whereas the vertical part is connected to the numerator value. EAR remains constant most of the times when the eye is open but falls to zero when the eye blinks.

$$EAR = (p2-p6+p3-p5*(p4-p1))/2$$

We need different varieties of packages like Scipy, Numpy, OpenCV etc to be installed in python.

Component Description

Face Detection

Finding the face is crucial since the eyes are located in the area of the face. This system uses an object detection algorithm to do this. To face and mark points on them, libraries like dlib, OpenCv, and imutils are utilised. To mark the points, we employ a 68-point face classifier. Following the identification of the face as our region of interest, it is cropped before moving on to the following stage, where the detection of the eye will take place.

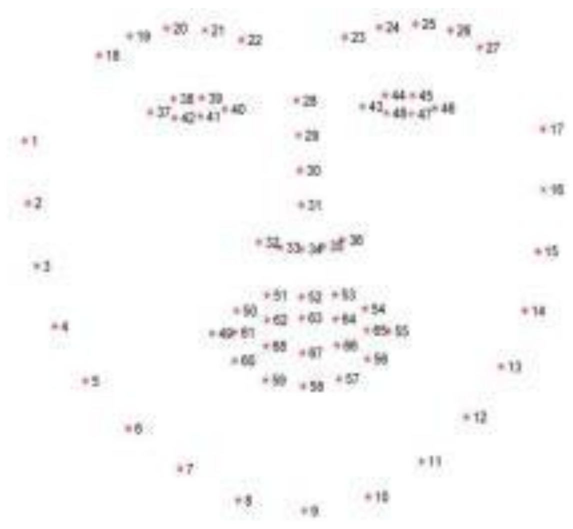


Figure 4. 68-point facial and mark [paper 11]

Eye Detection

The eye classifier, which has been downloaded from the OpenCV library, is obtained from the face area employing a similar technique as in eye detection. This minimises the amount of time needed for training and gets rid of the demanding procedures needed for evaluating performance. Once the eye region has been obtained, it is cut and moved to the complying with level, where the eye's state (closed/open) is determined.

Eye State Detection

When the eyes are detected, 6 points for each eye are obtained respectively. After that we will apply EAR (Eye Aspect Ratio) algorithm of both the eye whose computed average is considered.

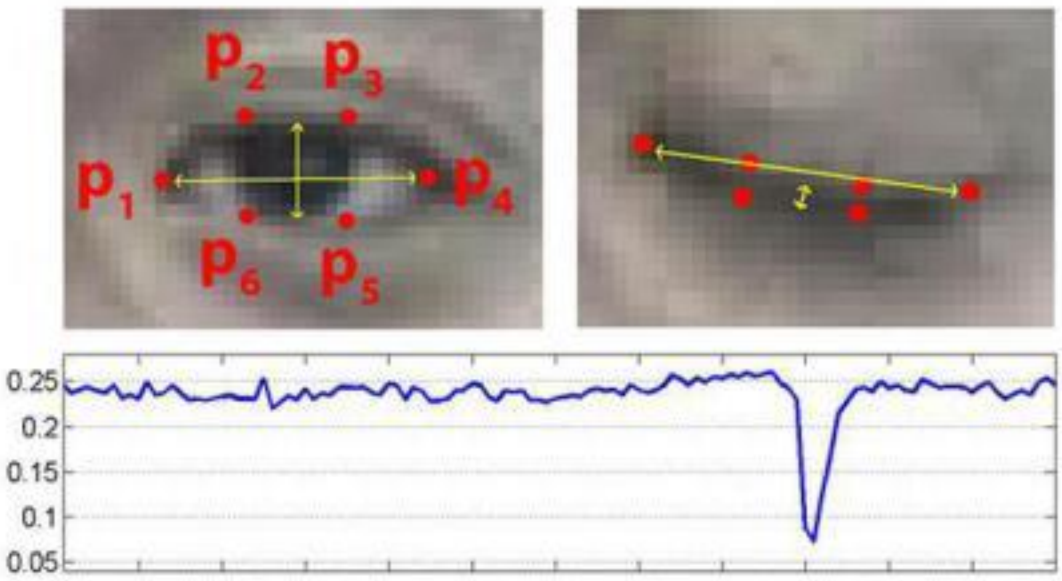


Figure 5. Eye state detection [paper 30]

$$EAR= (|p2-p6|+|p3-p5|)/2*(|p4-p1|)$$

As threshold varies from person to person computing is done on threshold EAR using– max (EAR)*0.75. When EAR goes below threshold, eye is supposed to be closed.

Drowsiness Alert Module

If the driver remains drowsy for 30 consecutive frames, surveillance is activated and an alarm is activated, until the driver get back to non-drowsy state.

Features :

The code uses OpencV to generate a live videostream and calculate the EAR in each individual frame, in real time.

Built-in functions in the dlib library, get_frontal_face_detector() and shape_predictor() are used to predict and detect the face and facial features (in this case, the eye).

- scipy.spatial library is used to compute the Euclidian distance between two points of the eye.
- argparse library is used to detect the path of the trained dataset in the dlib library
- VideoStream() is a built-in function in the cv2 library which launches the camera device passed in its arguments, which in this project is the webcam of the device.
- cvtColor() is also a built-in function of the cv2 library which is used to convert the frame capturd by the webcam to grayscale, as it is easier to detect facial features in grayscale.
- convexHull() is a built-in function of the cv2 library which draws convex arcs between two points passed as arguments.

Dataset :

We have used a trained dataset which is a standard dataset used in the dlib library and which works on the 68-point face detection concept.

4. Modelling and Implementation Details

4.1 Design Diagrams

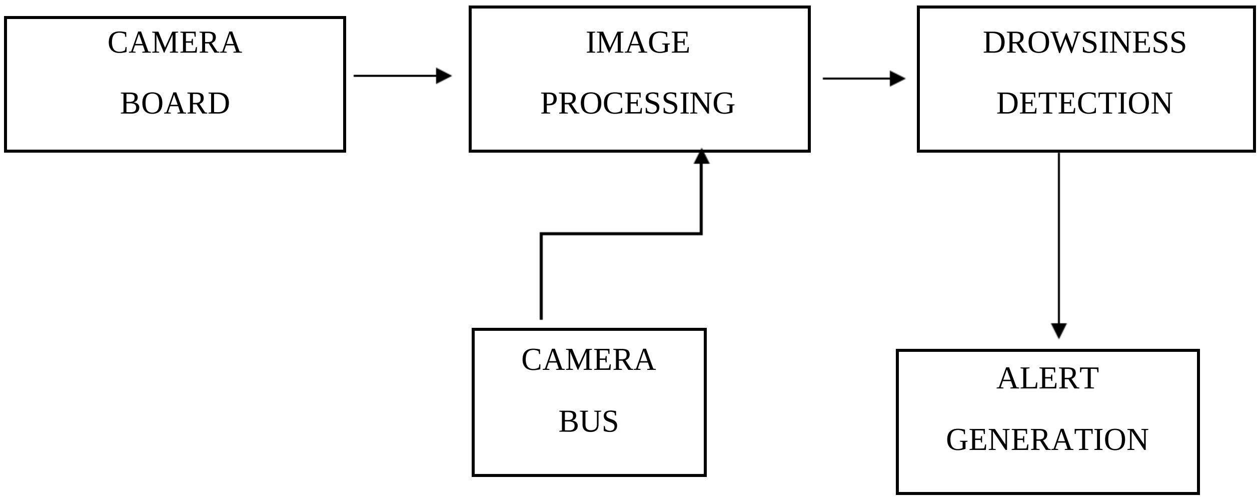


Figure 6. Basic Architecture

4.1.1 Use case Diagram

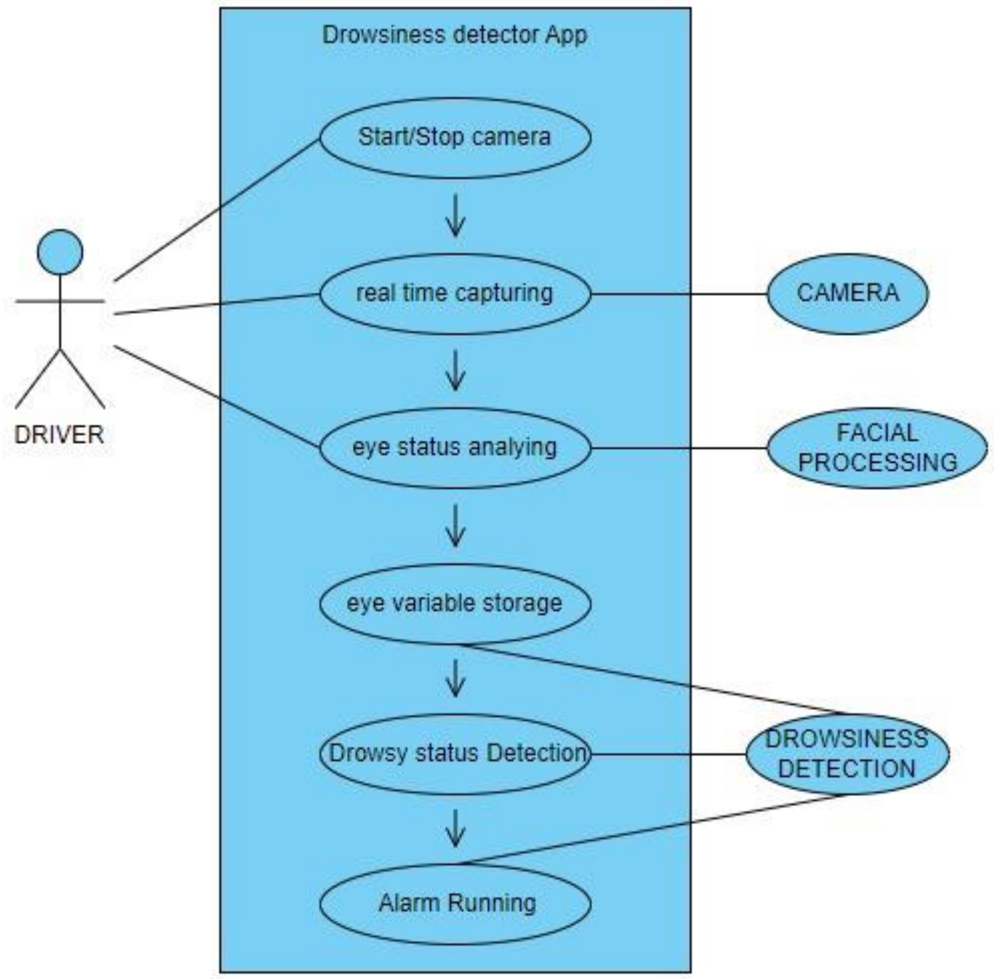


Figure 7. Use Case Diagram[visualparadigm.com]

4.1.2 Control Flow Diagram

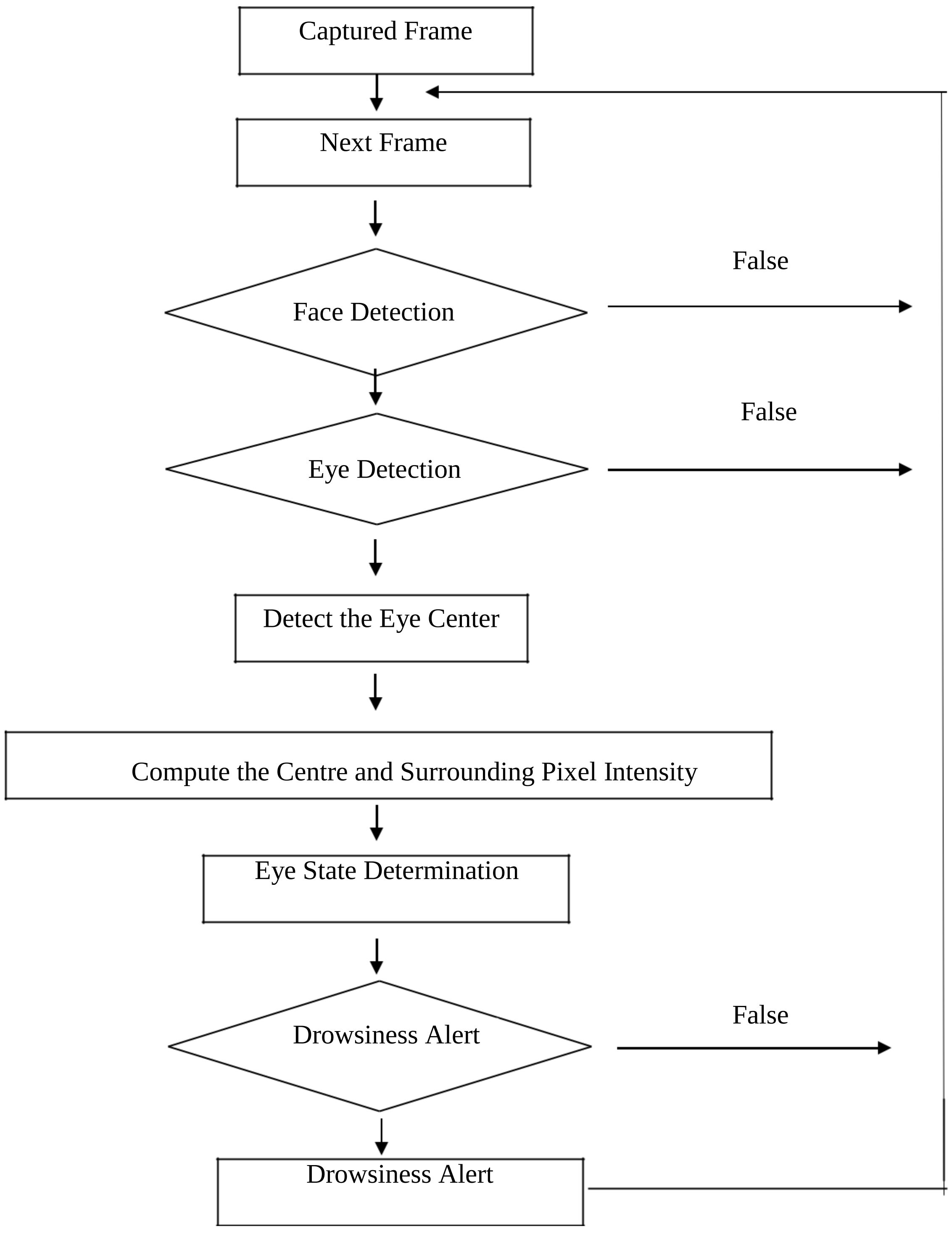


Figure 8. Control Flow Diagram

4.2 Implementation details and issues

The goal of this module is to process the captured video pictures. The eyes are first recognized , and then the face of the driver is located in the footage stream. Then, using the pixel intensity difference and a threshold value, the eye's condition will be determined. Six coordinates of the eye are taken into account by The EAR algorithm (Eye Aspect Ratio). The denominator half of the EAR algorithm's formula corresponds to horizontal eye markings, whereas the vertical part is connected to the numerator value. When the eye is open, EAR generally stays constant, but when the eye blinks, it zeros out.

$$EAR= (|p2-p6|+|p3-p5|)/2*(|p4-p1|)$$

We need different varieties of packages like Scipy, Numpy, Opencvetc to be installed in python.

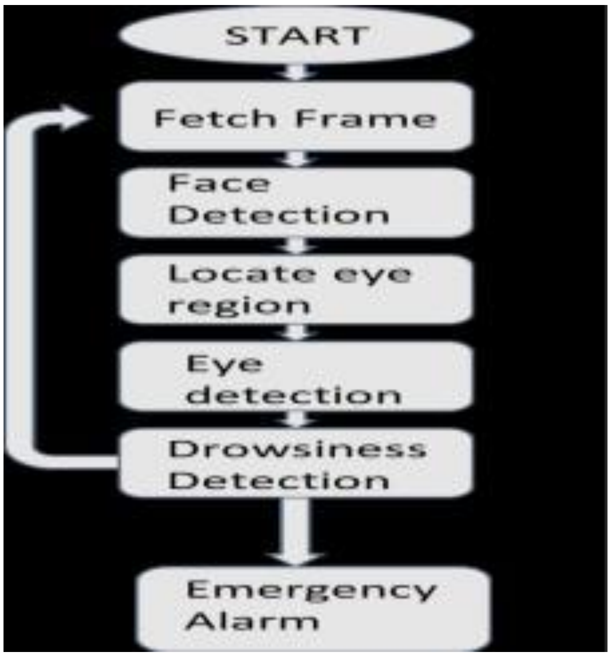
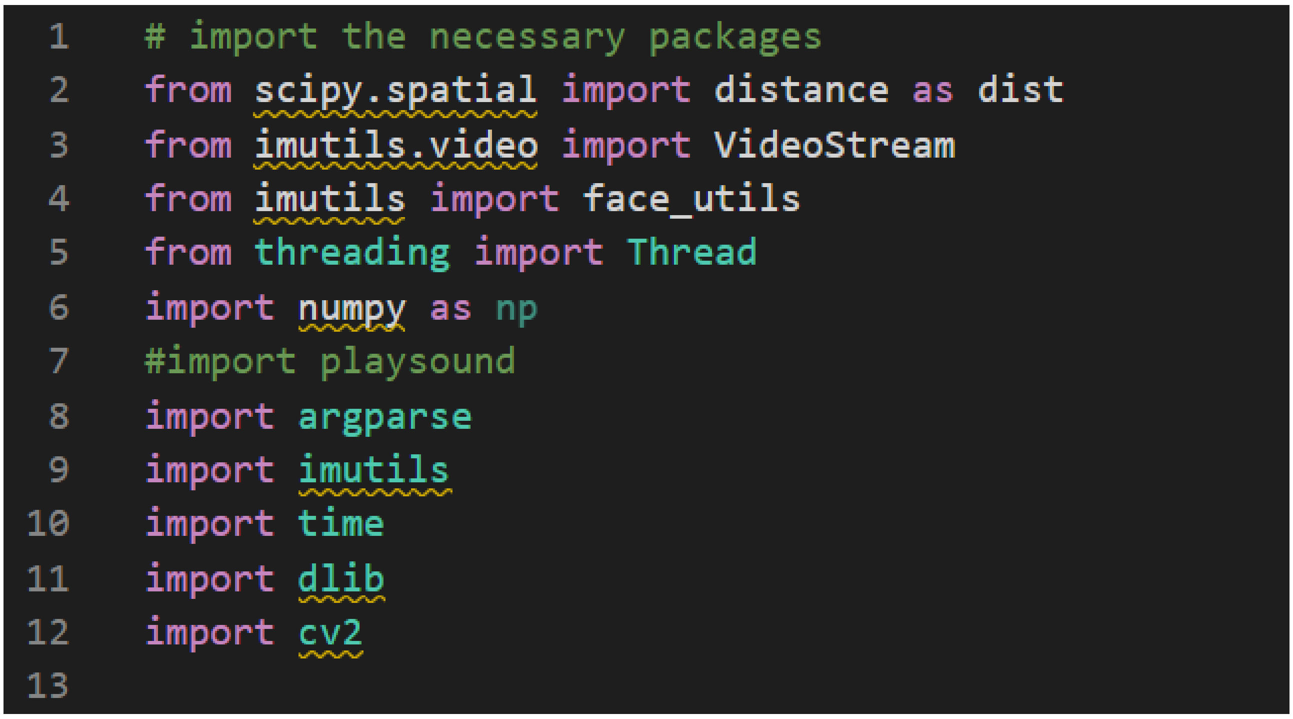


Figure 9. Process involved [paper 7]

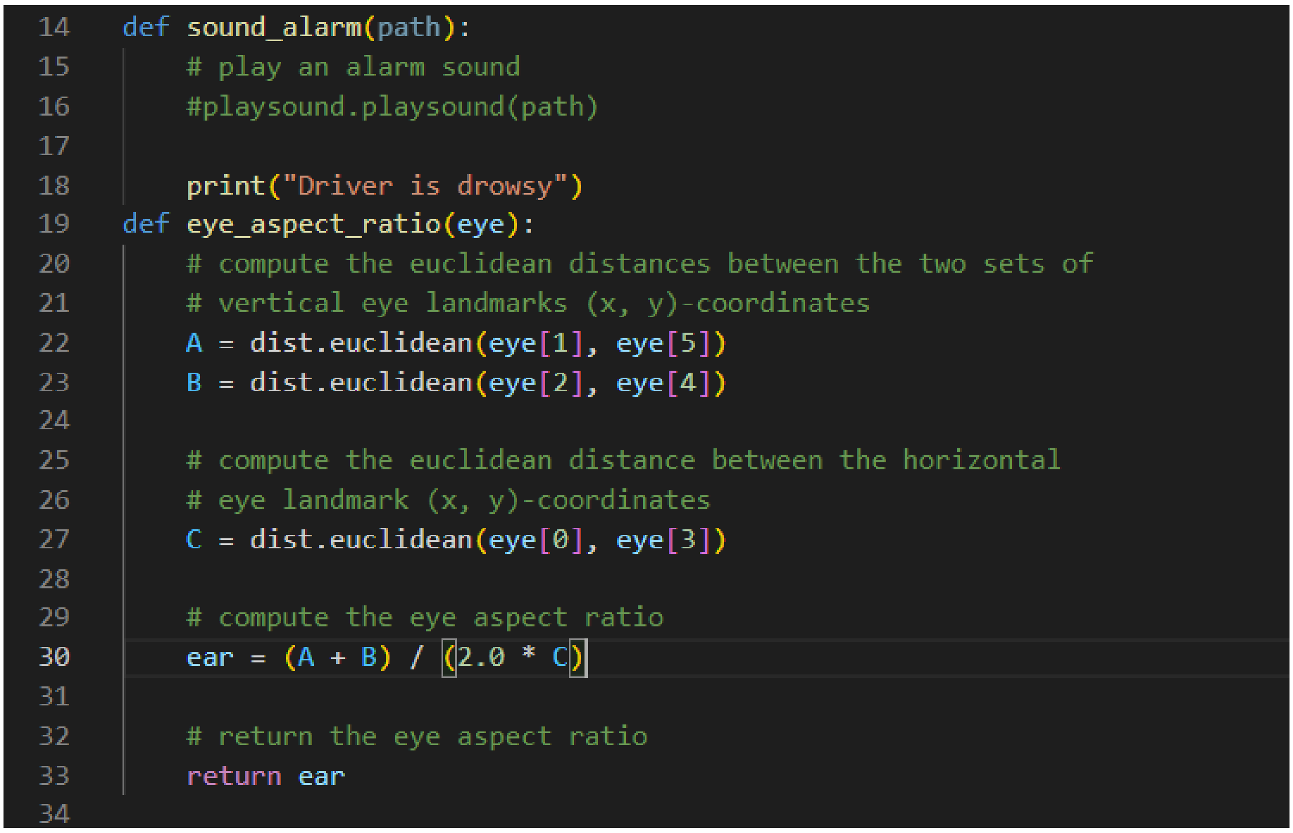


```python
# import the necessary packages
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
#import playsound
import argparse
import imutils
import time
import dlib
import cv2

```

Figure 10.Code explanation 1

- scipy.spatial.distance: This package provides functions for computing distances between objects in a space. It's commonly used in computer vision for calculating distances between image features.
- imutils.video: This library provides a simple interface for capturing video from cameras or files, and for processing video frames. It's built on top of the OpenCV library.
- imutils.face_utils: This module provides several utility functions for working with facial landmarks detected using the dlib library. It includes functions for calculating eye aspect ratio, mouth aspect ratio, and other facial landmarks.
- threading.Thread: This class provides a way to run Python code in separate threads. Threads are useful for running long-running tasks in the background while the main program continues to run.
- numpy: This is a popular numerical computing library for Python. It provides functions for working with arrays and matrices, and for performing mathematical operations on them.
- argparse: This library provides a way to parse command line arguments in a Python script. It's useful for providing user options and settings when running a script.
- imutils: This is a collection of utility functions for working with OpenCV. It includes functions for resizing images, rotating images, and applying masks.
- time: This module provides functions for working with time and dates in Python.
- dlib: This is a popular library for face detection, facial landmark detection, and other computer vision tasks. It includes pre-trained models for detecting faces and landmarks in images and video.

•	cv2: This is the Python binding for OpenCV, a popular computer vision library. It provides functions for image and video processing, such as reading and writing image files, capturing video from cameras, and applying



```python
def sound_alarm(path):
    # play an alarm sound
    #playsound.playsound(path)

    print("Driver is drowsy")
def eye_aspect_ratio(eye):
    # compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])

    # compute the euclidean distance between the horizontal
    # eye landmark (x, y)-coordinates
    C = dist.euclidean(eye[0], eye[3])

    # compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)

    # return the eye aspect ratio
    return ear
```

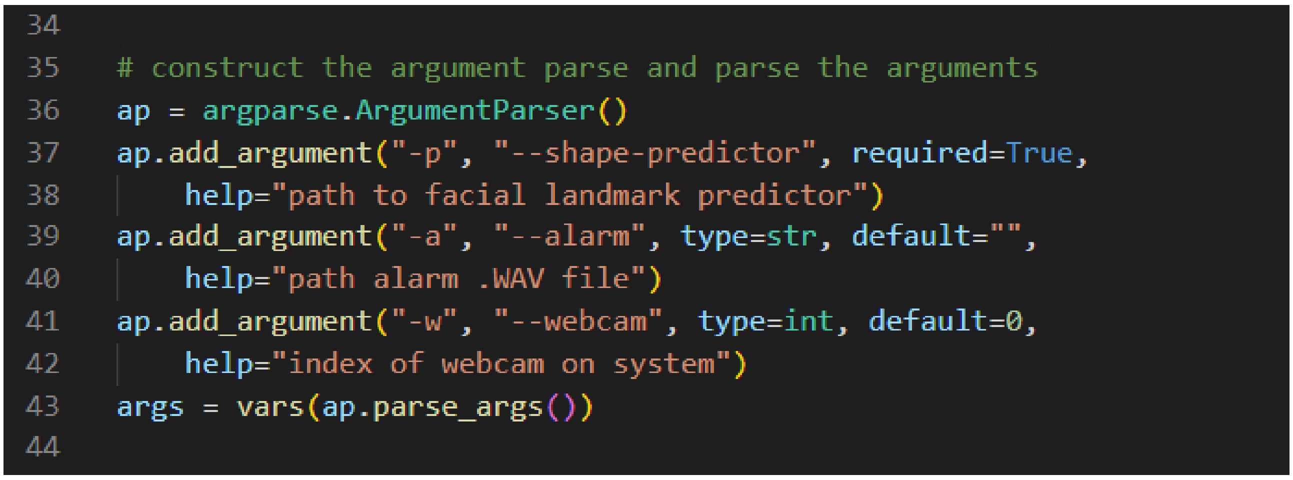
Figure 11.Code explanation 2

1. sound_alarm(path): This function only accepts one parameter, path, which specifies the location of the audio file to be played as the alarm. The actual sound playing is, however, commented out in this code. As an alternative, it merely outputs "Driver is drowsy" to the console. When the system notices that the driver is sleepy or not paying attention, it often invokes this function.

2. eye_aspect_ratio(eye): The above function accepts a single parameter eye, which is a list of six (x, y) coordinates corresponding to the locations of the landmarks that make up an eye's border (specifically, the points on the top, bottom, left, and right margins of the eye).

The function calculates the Euclidean distance between two sets of vertical eye landmarks (A) and the Euclidean distance between the two sets of horizontal eye landmarks (B). It then computes the Euclidean distance between the horizontal eye landmark and the midpoint between the top and bottom landmarks (C).

The eye aspect ratio is then calculated as the average ratio of the distance between the vertical landmarks (A and B) and the distance between the horizontal landmark and the midpoint (C). This calculation is based on research that has found that the eye aspect ratio is a good indicator of whether the eye is open or closed. The function returns the calculated eye aspect ratio.

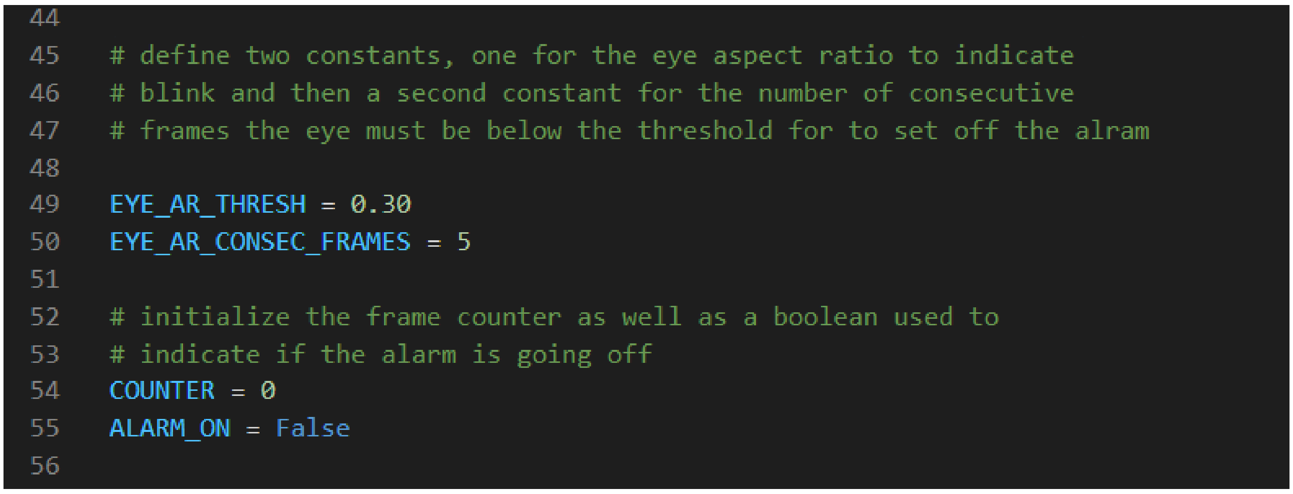


```python

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
    help="path to facial landmark predictor")
ap.add_argument("-a", "--alarm", type=str, default="",
    help="path alarm .WAV file")
ap.add_argument("-w", "--webcam", type=int, default=0,
    help="index of webcam on system")
args = vars(ap.parse_args())

```

Figure 12.Code explanation 3

1. ap = argparse.ArgumentParser(): This creates an instance of the ArgumentParser class, which is used to define and parse command-line arguments.
2. ap.add_argument("-p", "--shape-predictor", required=True, help="path to facial landmark predictor"): This adds an argument to the parser using the add_argument() method. The argument is defined with a short flag (-p) and a long flag (--shape-predictor) and is marked as required using required=True. The help argument provides a brief description of what the argument does. This argument is used to specify the path to the facial landmark predictor file.
3. Add another option to the parser using ap.add_argument("-a", "--alarm", type=str, default="", help="path alarm.WAV file"). The location to the audio file that will be played when the system detects sleepiness is specified using this option. The argument's default value is an empty string, hence it is not necessary because of this.
4. The parser is given a new argument with the aid of ap.add_argument("-w", "--webcam", type=int, default=0, and "index of webcam on system"). The index of the camera on the system that will be utilised to record video frames is specified using this option. If no argument is given, the option's default value of 0 implies that the first webcam discovered will be utilised.

5. args = vars(ap.parse_args()): This actually parses the command-line arguments using the parse_args() method, and stores the result in a dictionary object named args. The keys in this dictionary are the names of the arguments (e.g., "shape_predictor", "alarm", "webcam"), and the values are the corresponding values provided on the command line. This dictionary can be used to access the values of the arguments in the rest of the program.

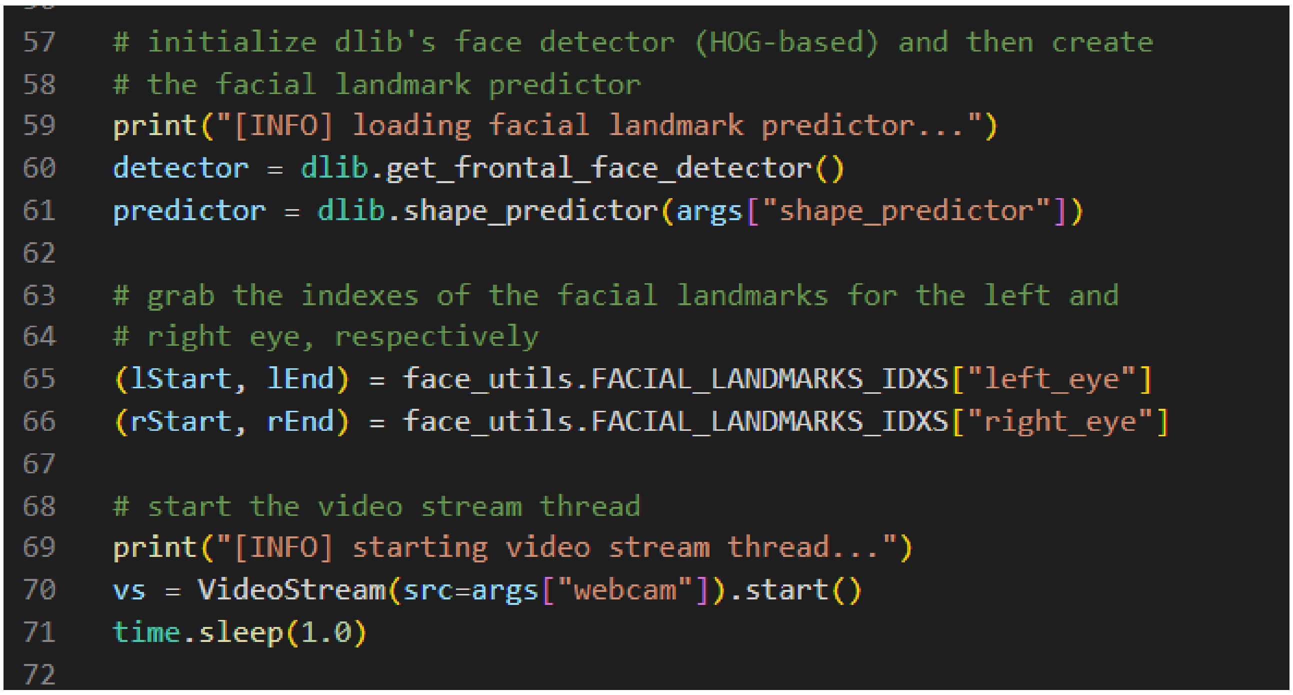


```

# define two constants, one for the eye aspect ratio to indicate
# blink and then a second constant for the number of consecutive
# frames the eye must be below the threshold for to set off the alram

EYE_AR_THRESH = 0.30
EYE_AR_CONSEC_FRAMES = 5

# initialize the frame counter as well as a boolean used to
# indicate if the alarm is going off
COUNTER = 0
ALARM_ON = False

```

Figure 13.Code explanation 4

1. EYE_AR_THRESH = 0.30: This defines a constant variable EYE_AR_THRESH with a value of 0.30. This constant is used to set a threshold for the eye aspect ratio, which is used to determine whether the driver's eyes are open or closed. If the eye aspect ratio falls below this threshold, it is an indication that the driver's eyes are closed or partially closed, which may indicate drowsiness.

2. EYE_AR_CONSEC_FRAMES = 5: This defines another constant variable EYE_AR_CONSEC_FRAMES with a value of 5. This constant is used to set the number of consecutive frames where the eye aspect ratio is below the threshold that must occur before the alarm is triggered. This is to avoid false alarms that may occur due to a temporary drop in the eye aspect ratio.
3. COUNTER = 0 and ALARM_ON = False: These lines initialize two variables. COUNTER is used to keep track of the number of consecutive frames where the eye aspect ratio is below the threshold. ALARM_ON is a boolean variable that is used to indicate whether the alarm is currently sounding or not. When the program starts, the counter is set to zero and the alarm is off (i.e., the boolean is set to False).



```
# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

# grab the indexes of the facial landmarks for the left and
# right eye, respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# start the video stream thread
print("[INFO] starting video stream thread...")
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)
```

Figure 14.Code explanation 5

1. print("[INFO] loading facial landmark predictor..."): This prints a message to the console indicating that the facial landmark predictor is being loaded.
2. detector = dlib.get_frontal_face_detector(): This initializes the dlib face detector that is used to detect faces in the video stream.
3. predictor = dlib.shape_predictor(args["shape_predictor"]): This imports the visage features predictor using the path to the predictor file that was specified as a command-line argument. This predictor is used to detect the locations of various facial landmarks, including the eyes.
4. (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"] and (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]: These lines set up the indexes for the left and right eye landmarks using the face_utils.FACIAL_LANDMARKS_IDXS dictionary that maps facial landmark names to their corresponding indexes.
5. print("[INFO] starting video stream thread..."): This prints a message to the console indicating that the video stream thread is being started.
6. vs = VideoStream(src=args["webcam"]).start(): This initializes the video stream using the index of the webcam on the system that was specified as a command-line argument. The VideoStream class from the imutils.video module is used for this.
7. time.sleep(1.0): This line adds a small delay of one second to allow the video stream to start up and stabilize.

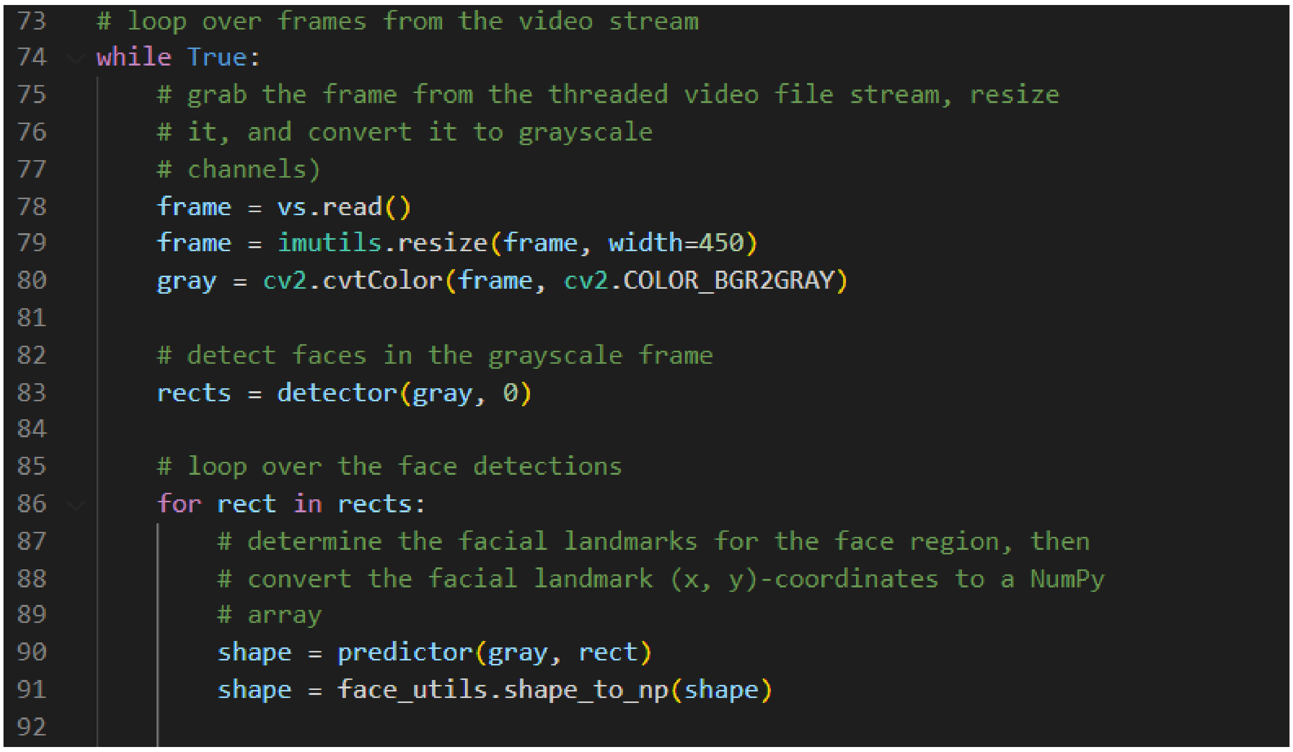
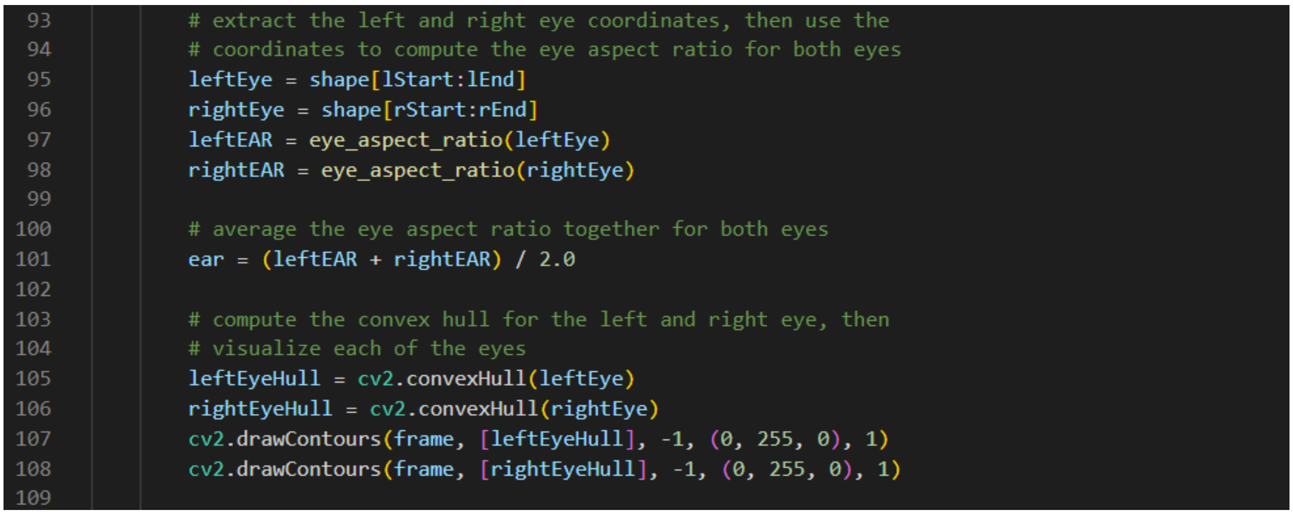

```python
# loop over frames from the video stream
while True:
    # grab the frame from the threaded video file stream, resize
    # it, and convert it to grayscale
    # channels)
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces in the grayscale frame
    rects = detector(gray, 0)

    # loop over the face detections
    for rect in rects:
        # determine the facial landmarks for the face region, then
        # convert the facial landmark (x, y)-coordinates to a NumPy
        # array
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)

```

Figure 15.Code explanation 6

1. while True:: This starts an infinite loop that runs until the program is terminated.
2. frame = vs.read(): This reads a frame from the video stream using the VideoStream object's read() method and stores it in the frame variable.
3. frame = imutils.resize(frame, width=500): This resizes the frame to a width of 450 pixels using the imutils.resize() function.
4. gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY): This converts the resized frame to grayscale using the cv2.cvtColor() function from the OpenCV library.
5. rects = detector(gray, 0): This detects faces in the grayscale frame using the dlib face detector that was initialized earlier. The detector returns a list of rectangles that contain the detected faces.
6. for rect in rects:: This starts a loop over the detected faces.
7. shape = predictor(gray, rect): This uses the facial landmark predictor that was loaded earlier to detect the locations of various facial landmarks, including the eyes, in the current face. The predictor() method takes in the grayscale frame and the rectangle that encloses the face, and returns a set of facial landmark coordinates.
8. shape = face_utils.shape_to_np(shape): This converts the detected facial landmark coordinates to a NumPy array for easier processing. The face_utils module contains various utility functions for working with facial landmarks.

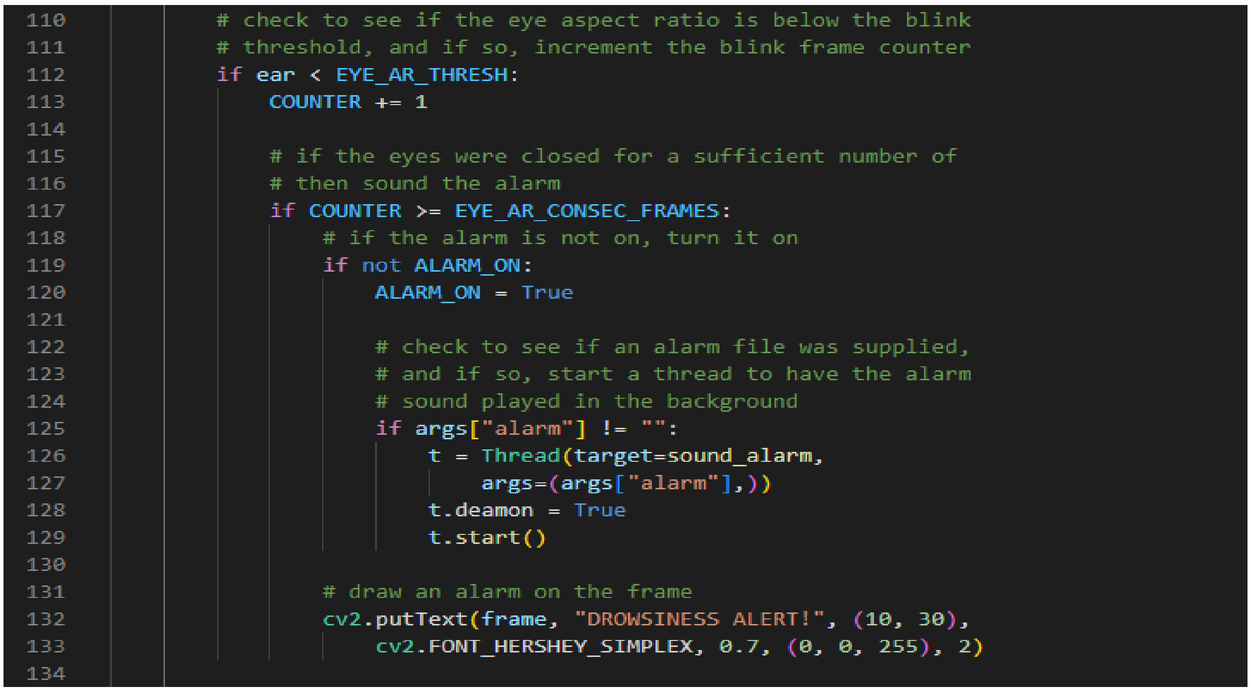


```
        # extract the left and right eye coordinates, then use the
        # coordinates to compute the eye aspect ratio for both eyes
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)

        # average the eye aspect ratio together for both eyes
        ear = (leftEAR + rightEAR) / 2.0

        # compute the convex hull for the left and right eye, then
        # visualize each of the eyes
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

```

Figure 16.Code explanation 7

This code block extracts the coordinates of the left and right eye landmarks from the facial landmark detection output. Then, it computes the eye aspect ratio (EAR) for both eyes using the function defined earlier.

Next, it averages the EAR values of both eyes to get a single EAR value for the frame. Then, it computes the convex hull of the left and right eye landmarks and visualizes them on the frame using the cv2.drawContours() function. The convex hull is the smallest convex polygon that can contain all the eye landmarks, and it is used to determine if the eyes are closed or open.



```
        # check to see if the eye aspect ratio is below the blink
        # threshold, and if so, increment the blink frame counter
        if ear < EYE_AR_THRESH:
            COUNTER += 1

            # if the eyes were closed for a sufficient number of
            # then sound the alarm
            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                # if the alarm is not on, turn it on
                if not ALARM_ON:
                    ALARM_ON = True

                    # check to see if an alarm file was supplied,
                    # and if so, start a thread to have the alarm
                    # sound played in the background
                    if args["alarm"] != "":
                        t = Thread(target=sound_alarm,
                            args=(args["alarm"],))
                        t.deamon = True
                        t.start()

                # draw an alarm on the frame
                cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

```

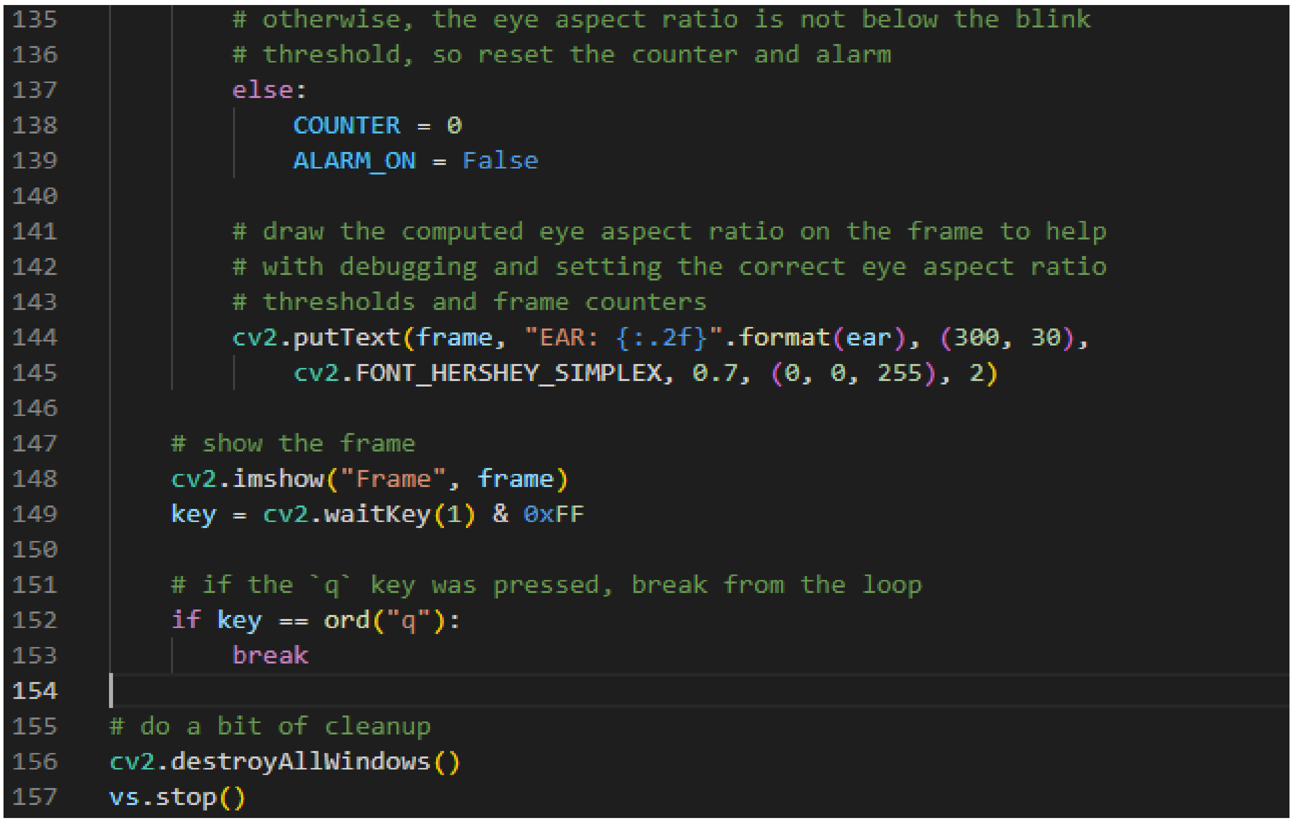
Figure 17.Code explanation 8

This code checks whether the eye aspect proportion is below a certain threshold EYE_AR_THRESH, which denotes an eye blink. The user is warned that they could be feeling sleepy if the eye is closed for a sufficient number of consecutive frames (EYE_AR_CONSEC_FRAMES).

The if ear < EYE_AR_THRESH: statement checks if the eye aspect ratio is below the blink threshold. If it is, then the COUNTER variable is incremented.

If the counter reaches the EYE_AR_CONSEC_FRAMES threshold, then an alarm is sounded by calling the sound_alarm function and setting the ALARM_ON variable to True. If an alarm sound file is provided, a new thread is started to play the sound in the background.

Finally, a text message "DROWSINESS ALERT!" is drawn on the video frame using OpenCV's cv2.putText() function.



```
            # otherwise, the eye aspect ratio is not below the blink
            # threshold, so reset the counter and alarm
            else:
                COUNTER = 0
                ALARM_ON = False

            # draw the computed eye aspect ratio on the frame to help
            # with debugging and setting the correct eye aspect ratio
            # thresholds and frame counters
            cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        # show the frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break

    # do a bit of cleanup
    cv2.destroyAllWindows()
    vs.stop()
```

Figure 18.Code explanation 9

This code block is responsible for displaying the output frame with relevant text overlay and listening for the "q" key to be pressed to end the program. It also performs cleanup operations before exiting.

The computed eye aspect ratio is shown on the frame using OpenCV's cv2.putText() function after the eye aspect proportion computation and eye blink recognition. The final step is to use cv2.imshow() to display the resultant frame.

The cv2.waitKey(1) function waits for a key to be pressed and returns the key code. If the key pressed is "q" (ord("q") == 113), then the program exits the loop and continues to the cleanup code. The cv2.waitKey() function is required to ensure that the OpenCV window is responsive to user input.

Finally, the OpenCV window is destroyed using cv2.destroy All Windows(), and the VideoStream object is stopped using its stop() method.

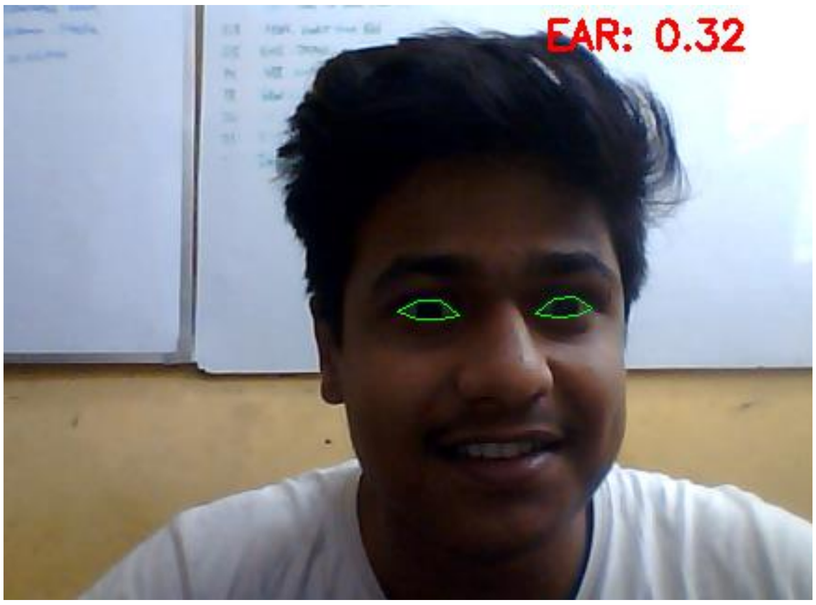


Figure 19. No alert while EAR is greater than threshold

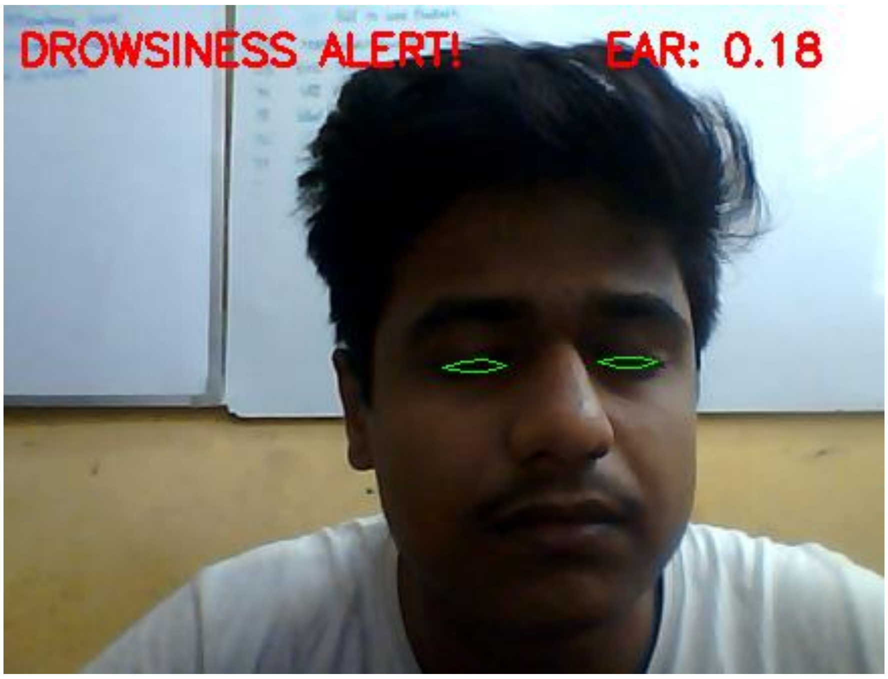
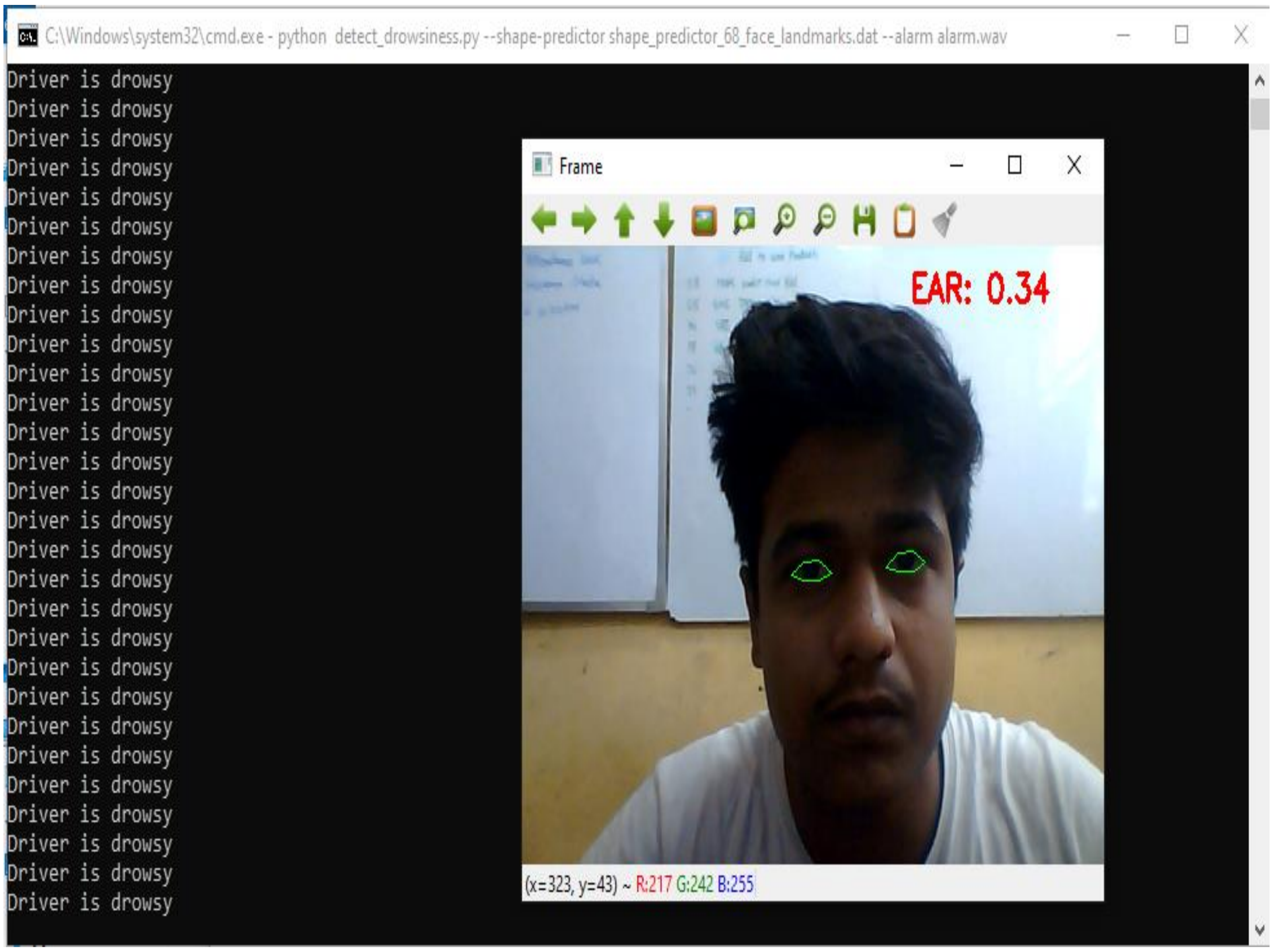
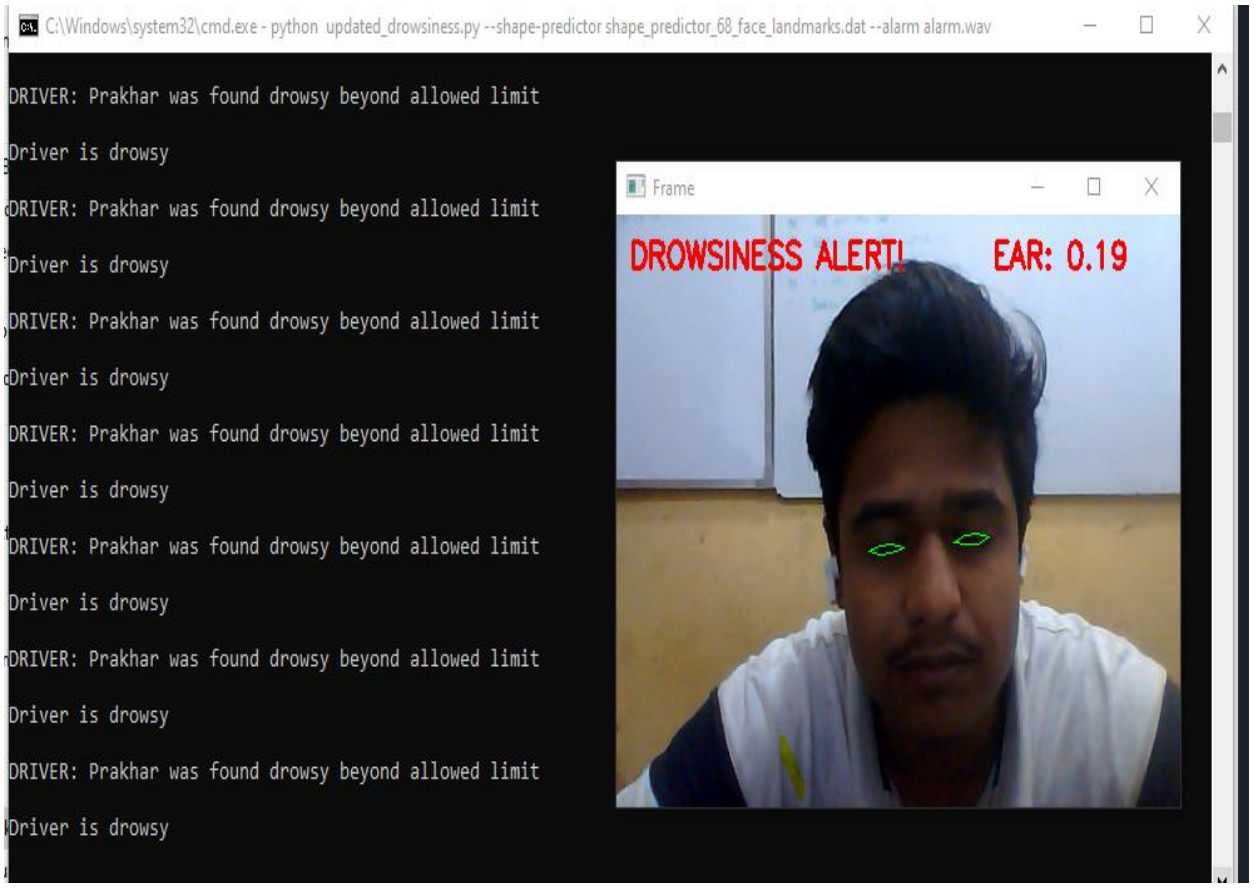


Figure 20. Alert while EAR is less than threshold



```
C:\Windows\system32\cmd.exe - python  detect_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
Driver is drowsy
```

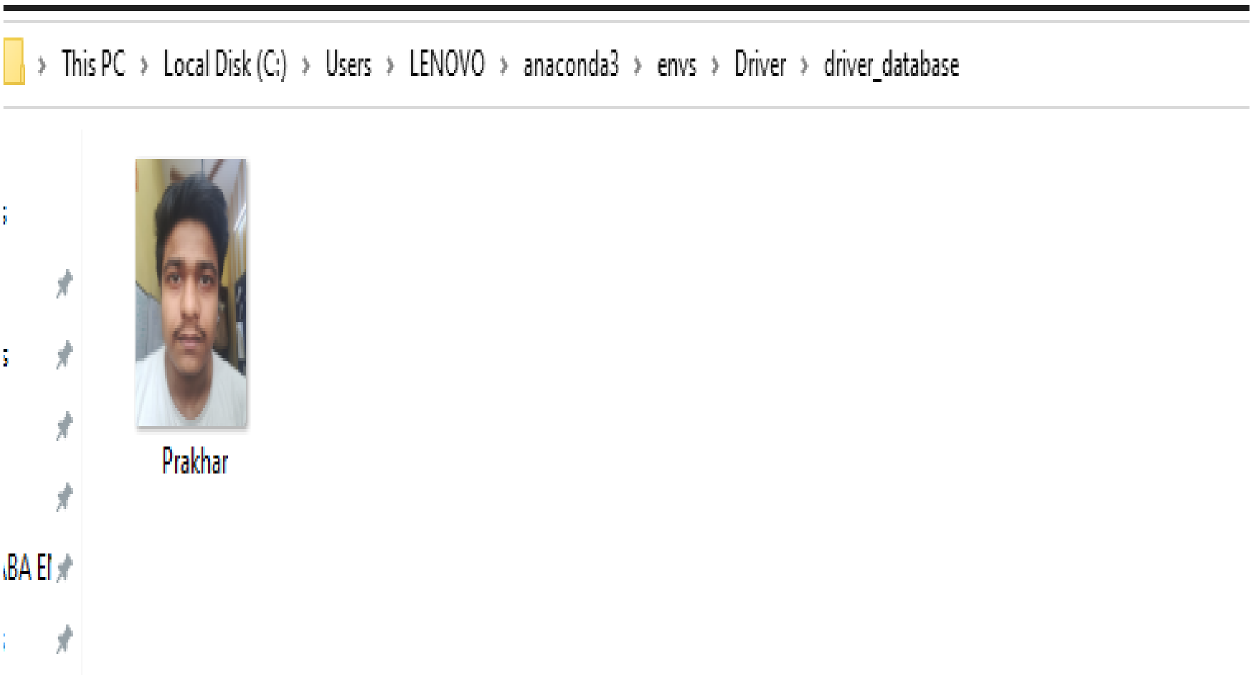
Figure 21. Alert while EAR is less than threshold on command screen



C:\Windows\system32\cmd.exe - python updated_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav

```
DRIVER: Prakhar was found drowsy beyond allowed limit
Driver is drowsy
DRIVER: Prakhar was found drowsy beyond allowed limit
Driver is drowsy
DRIVER: Prakhar was found drowsy beyond allowed limit
Driver is drowsy
DRIVER: Prakhar was found drowsy beyond allowed limit
Driver is drowsy
DRIVER: Prakhar was found drowsy beyond allowed limit
Driver is drowsy
DRIVER: Prakhar was found drowsy beyond allowed limit
Driver is drowsy
DRIVER: Prakhar was found drowsy beyond allowed limit
Driver is drowsy
```

Figure 22. Alert while EAR is less than threshold using database



This PC › Local Disk (C:) › Users › LENOVO › anaconda3 › envs › Driver › driver_database

Figure 23. Database Driver

Advantages

1. Monitors real time drowsiness of the driver.
2. Implemented as a web application.
3. Easy to use, as it can be used by a common person also.
4. Helps in reducing road accidents during night

Algorithm

1. Set up a camera that watches an ongoing collection of faces first.
2. In the event that a face is discovered, facial markings are used to locate and extract the eye area.
3. To detect whether the eyes are closed, the eye dimension ratio will now be calculated for a specific eye area.
4. An alarm will sound to wake up the driver if the eye aspect ratio (EAR) shows that the driver's eyes have been closed for a long enough period of time.

Face and eye detection:

Steps:

1. All required libraries like imutils, numpy are imported.
2. Constants are provided for the number of consecutive frames the eye must be below the threshold for the alert to sound and for the eye aspect ratio to signal blink.
3. The calculation of the Euclidean distance between both sets of vertical eye markers (x, y)-coordinates.
4. Eye aspect ratio is calculated after calculating the Euclidean distance.
5. The facial landmark predictor and face detector in Dlib are initialised.
6. Assessment of the left and right eye's respective facial marker indexes.
7. The video stream thread is started and the face is detected.
8. After face detection, the facial landmarks for the face region are determined.
9. The convex hull for the left and right eye is computed and visualized for each of the eyes.
10. Values are checked to see if the eye aspect ratio is below the blinking threshold, and if so, the blink frame counter is incremented.
11. If the value exceeds the threshold, alarm is turned on.
12. If not, the eye aspect ratio is not below the blink threshold and so the counter and the alarm is reset.
13. The obtained frame is displayed.
14. All the values are re initialized for the next frame.

Implementation Issues

- Varying results due to demographically changing eye aspect ratio threshold
- Inability to produce results in case of eye impairment
- The non-connectivity of the designed system from the vehicle’s engine.
- Inability for night vision detection.

4.3 Risk Analysis and Mitigation

Table I. Risk Analysis

Risk	Classification	Risk Description	Risk Area	Probability (P)	Impact (I)
1	Hardware	Hardware incapability, such as low RAM, processor, memory, or camera module	Performance, Time	Medium	High
2	Multi-tenancy	All users share the same physical architecture, leading to potential security issues	Security	Low	High
3	Security	Issues related to authentication, authorization, and access control	User, Scope, Time	High	High
4	Hardware	Processor performance issues	Performance, Time	Low	High
5	Ownership	Organization owning the system, leading to potential security risks	Security Performance, Time	High	Low
6	Environment	Performance issues related		High	Medium
	Environment	to Windows environment Performance issues related to alarms and cameras	Performance	High	High
8	Related Personnel	Incompetent skills of personnel	Time	High	High
9	Related Personnel	Irregularity of personnel	Time	Medium	High

Table II. Mitigation Plan

Risk	**Mitigation Plan**
Hardware	devices and cameras
Security	Secure connection must be established. Camera’s and alarms must not be mishandled.
Personnel Related	Irregularities should be handled.

5. TESTING

5.1 Testing Plan

Table III. Testing Plan

Type of Test	Has it been Performed	Explanations	Software Components
Requirement Testing	Yes	Requirements specification must contain all the requirements that are to be solved by our system.	Manual work, need to plan out all the software requirements, time needed to develop, technology to be used etc.
Unit	Yes	Testing technique using which individual modules are tested to determine if there are any issues, by the developer himself.	Manual check is required.
Integration	Yes	Testing where individual components are combined and tested as a group.	Compiling full part of the code (face detection & authentication) and testing it together.
Performance	Yes	Testing to evaluate the input where the best and most optimal output is yielded by the system.	Protocols and testing results used ensure this.
Stress	No	Not needed	N/A
Compliance	No	Not needed	N/A
Security	No	Not needed	There are no security issues.
Load	No	Not needed	Does not depend on multiple user access.
Volume	No	Not needed	Executes in time.

5.2 Component decomposition and type of testing required

Table IV. Component Decomposition and Identification of Tests required

S. No.	**List of various modules that required testing**	**Type of Testing required**	**Techniques for writing test cases**
1	Eye detection	Requirement, Unit, Performance, Integration	White box
2	Face Authentication		White box
3	Camera		White box
4	Alarm		

5.3 Test Cases

Table V. Test cases for components

Test Case ID	**Input**	**Expected Output**	**Status**
1	Face detection of an individual aged from 3-80 years	Accuracy	Pass
2	Face detection with Spectacles	Accuracy	Pass
3	Face with eye impairment	Accuracy	Fail
4	Face detection at night	Not Accurate	Pass

5.4 Error and Exception Handling

Table VI. Type of debugging technique used

Test Case ID	**Test case for Components**	**Debugging Technique**
1	Eye detection with eye impairment	Retina scanner technology can be implemented
2	More than one faces in eye detection	More efficient algorithms can be facilitated
3	Alarm is raised if drowsy in a static vehicle.	Connection to the engine
4	Unauthenticated driver access	Connection to the engine
5	Results are not established in dark	Night vision enabled cameras

5.5 Limitations of the solution

Although the system performed well and produced good results but main limitation of this system is the cost. As better cameras are required for providing better and accurate results.

6. FINDINGS AND CONCLUSIONS

6.1 Findings

The system was found efficiently detecting fatigue measuring the eye aspect ratio and subsequently authenticating the driver. In cases where drowsiness was detected an alarm was formulated in addition the face of the assigned driver is also authenticated accurately to ensure safety and security of the vehicles.

6.2 Conclusion

Now a day, driver drowsiness is one of the major reasons for road accidents and in order to reduce this there is a series need to develop safety measures to prevent this. Drowsiness can arise due to many reasons like long working hours, minimum visibility during night, distractions, alcohol consumption etc.

Not only this, constant speed of the vehicle for longer duration can also lead to many road accidents. Keeping these reasons in mind we focused to make a system which will detect whether our driver is drowsy or not. If any situation like drowsiness comes into picture, then our model will detect it and a warning in the form of alarm will be raised to the particular driver.

Initially our system has used python libraries and detected drowsiness but later the system will be enhanced by using Whale Optimization Algorithm for improving the efficiency of neural networks.

6.3 Future Work

Environmental lighting may be handled in order to enhance this technology by introducing a module that can measure the illumination levels and alter the threshold value for blink detection correspondingly. More potent cameras with greater frame rates can be used to improve the system's effectiveness and add more sleepiness measure parameters. The aforementioned devices will cost a bit extra, though. By locking the steering and brakes and sounding a theft alert, access to a foreign organisation can be restricted for better authentication. A transmission module can be integrated to convey the driving status information in real-time to the appropriate authorities, improving control and increasing driver situation monitoring. An active technique of video recording, such as an infrared camera instead of visible light dependant cameras, can be used to improve the system's performance at night or in locations with low lighting levels.

Future research should cover more advanced behavioural patterns in front of the surveillance device that signify driver drowsiness. The advancements in image processing has enhanced the facial recognition capabilities making the systems more definitive and better suited for applications in security and protection systems. This system differs from efficient algorithms in that it processes data efficiently, reliably quickly, and accurately. The technology can determine whether the eyes are open or closed while it is monitoring. When the eyes are closed for an extended period of time, a warning signal is sent and a punishment is levied. Similar ones were planned out. The improvement and development of this technology might potentially save a large number of lives.. Advance Vehicle Safety System Using Face Detection will cause a significant decline in the number of accidents and ensure driver safety.

REFERENCES

[1] K. Cen, “Study of viola-jones real time face detector,” 2018.

[2] R. Feraund, O. J. Bernier, J.-E. Viallet, and M. Collobert, “A fast and accurate face detector based on neural networks,” IEEE Transactions on pattern analysis and machine intelligence, vol. 23, no. 1, pp. 42–53, 2001.

[3] D. Huang, C. Shan, M. Ardabilian, Y. Wang, and L. Chen, “Local binary patterns and its application to facial image analysis: a survey,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 6, pp. 765–781, 2011.

[4] J. Krajewski, D. Sommer, U. Trutschel, D. Edwards, and M. Golz, “Steering wheel behavior based estimation of fatigue,” 2009.

[5] A. M. Malla, P. R. Davidson, P. J. Bones, R. Green, and R. D. Jones, “Automated video-based measurement of eye closure for detecting behavioral microsleep,” in 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology. IEEE, 2010, pp. 6741–6744.

[6] V. Saini and R. Saini, “Driver drowsiness detection system and techniques: a review,” International Journal of Computer Science and Information Technologies, vol. 5, no. 3, pp. 4245–4249, 2014.

[7] T. Soukupova and J. Cech, “Eye blink detection using facial landmarks,” in 21st Computer Vision Winter Workshop, RimskeToplice, Slovenia, 2016.

[8] S. Mehta, S. Dadhich, S. Gumber, and A. Jadhav Bhatt, “Real-time driver drowsiness detection system using eye aspect ratio and eye closure ratio,” Available at SSRN 3356401, 2019.

[9] P. Bharadwaj, A. CN, T. S. Patel, K. BR et al., “Drowsiness detection and accident avoidance system in vehicles,” 2019.

[10] R. P. Kumar, M. Sangeeth, K. Vaidhyanathan, and M. A. Pandian, “Traffic sign and drowsiness detection using open-cv,” TRAFFIC, vol. 6, no. 03, 2019.

[11] W. Deng and R. Wu, "Real-time driver-drowsiness detection system using facial features," IEEE Access, vol. 7, pp. 118 727–118 738, 2019.

[12] B.-L. Lee, B.-G. Lee, and W.-Y. Chung, "Standalone wearable driver drowsiness detection system in a smartwatch," IEEE Sensors journal, vol. 16, no. 13, pp. 5444–5451, 2016.

[13] A. Mittal, K. Kumar, S. Dhamija, and M. Kaur, "Head movement based driver drowsiness detection: A review of state-of-art techniques," in 2016 IEEE International Conference on Engineering and Technology (ICETECH). IEEE, 2016, pp. 903–908.

[14] T. Wang and P. Shi, "Yawning detection for determining driver drowsiness," in Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, 2005. IEEE, 2005, pp. 373–376.

[15] M. Babaeian, N. Bhardwaj, B. Esquivel, and M. Mozumdar, "Real time driver drowsiness detection using a logistic-regression-based machine learning algorithm," in 2016 IEEE Green Energy and Systems Conference (IGSEC). IEEE, 2016, pp. 1–6.

[16] B. G. Pratama, I. Ardiyanto, and T. B. Adji, "A review on driver drowsiness based on image, bio-signal, and driver behavior," in 2017 3rd International Conference on Science and Technology-Computer (ICST). IEEE, 2017, pp. 70–75.

[17] M. Awais, N. Badruddin, and M. Drieberg, "A hybrid approach to detect driver drowsiness utilizing physiological signals to improve system performance and wearability," Sensors, vol. 17, no. 9, p. 1991, 2017.

[18] M. Chai et al., "Drowsiness monitoring based on steering wheel status," Transportation research part D: transport and environment, vol. 66, pp. 95–103, 2019.

[19] B.-H. Jang, I.-H. Park, H.-D. Nam, and K.-H. Kim, "A study on the relationship between posture recognition and drowsy driving," The Transactions of The Korean Institute of Electrical Engineers, vol. 67, no. 7, pp. 934–939, 2018.

[20] M. A. Assari and M. Rahmati, "Driver drowsiness detection using face expression recognition," in 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA). IEEE, 2011, pp. 337–341.

[21] W. Zhang, B. Cheng, and Y. Lin, "Driver drowsiness recognition based on computer vision technology," Tsinghua Science and Technology, vol. 17, no. 3, pp. 354–362, 2012.

[22] Z. Li, L. Chen, J. Peng, and Y. Wu, “Automatic detection of driver fatigue using driving operation information for transportation safety,” Sensors, vol. 17, no. 6, p. 1212, 2017.

[23] F. Zhang, J. Su, L. Geng, and Z. Xiao, “Driver fatigue detection based on eye state detect in 2017 International Conference on Machine Vision and Information Technology (CMVIT). IEEE, 2017, pp. 105–110.

[24] A. Sayeed and S. A. Sadi, “Driver drowsiness detection using face monitoring and pressure measurement,” Research & Reviews: A Journal of Embedded System & Applications, vol. 5, no. 3,
pp. 12–18, 2018.

[25] K. Dwivedi, K. Biswaranjan, and A. Sethi, “Drowsy driver detection using representation learning,” in 2014 IEEE International Advance Computing Conference (IACC). IEEE, 2014, pp. 995– 999.

[26] I. Daza, L. Bergasa, S. Bronte, J. Yebes, J. Almaz´an, and R. Arroyo, “Fusion of optimized indicators from advanced driver assistance systems (adas) for driver drowsiness detection,” Sensors, vol. 14, no. 1, pp. 1106– 1131, 2014.

[27] S. Park, F. Pan, S. Kang, and C. D. Yoo, \Driver drowsiness detection sys-tem based on feature representation learning using various deep networks," in Asian Conference on Computer Vision. Springer, 2016, pp. 154-164.

[28] G. Li and W.-Y. Chung, “Detection of driver drowsiness using wavelet analysis of heart rate variability and a support vector machine classifier,” Sensors, vol. 13, no. 12, pp. 16 494–16 511, 2013.

[29] L.-B. Chen, W.-J. Chang, J.-P. Su, J.-Y. Ciou, Y.-J. Ciou, C.-C. Kuo, and K. S.-M. Li, \A wearable-glasses-based drowsiness-fatigue-detection system for improving road safety," in 2016 IEEE 5th Global Conference on Consumer Electronics. IEEE, 2016, pp. 1-2.

[30] A. Asthana, S. Zafeoriou, S. Cheng, and M. Pantic. Incremental face alignment in the wild. In Conference on Computer Vision and Pattern Recognition, 2014. 1, 2, 3, 4, 5, 7.

9919103206_2_AUTOMATED DRIVER DROWSINESS DETECTION SYSTEM

ORIGINALITY REPORT

19%	18%	11%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	3%
2	www.pyimagesearch.com Internet Source	2%
3	de.slideshare.net Internet Source	1%
4	marswebsolutions.files.wordpress.com Internet Source	1%
5	www.ijraset.com Internet Source	1%
6	github.com Internet Source	1%
7	Submitted to Indian Institute of Technology Guwahati Student Paper	1%
8	www.ijres.org Internet Source	1%
	link.springer.com	

9	Internet Source	1%
10	Submitted to University of Sunderland Student Paper	1%
11	Boon-Leng Lee, Boon-Giin Lee, Wan-Young Chung. "Standalone Wearable Driver Drowsiness Detection System in a Smartwatch", IEEE Sensors Journal, 2016 Publication	1%
12	sciences.uodiyala.edu.iq Internet Source	1%
13	Submitted to Sogang University Student Paper	<1%
14	Submitted to University of Glamorgan Student Paper	<1%
15	Md. Yousuf Hossain, Fabian Parsia George. "IOT Based Real-Time Drowsy Driving Detection System for the Prevention of Road Accidents", 2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), 2018 Publication	<1%
16	www.jetir.org Internet Source	<1%
17	www.coursehero.com Internet Source	<1%

18	www.ijsrd.com Internet Source	<1 %
19	www.slideshare.net Internet Source	<1 %
20	worldwidescience.org Internet Source	<1 %
21	Submitted to Chandigarh University Student Paper	<1 %
22	Submitted to Glyndwr University Student Paper	<1 %
23	"ACIT 2021 Conference Proceedings", 2021 22nd International Arab Conference on Information Technology (ACIT), 2021 Publication	<1 %
24	www.koreascience.or.kr Internet Source	<1 %
25	www.jatit.org Internet Source	<1 %
26	cse.anits.edu.in Internet Source	<1 %
27	Submitted to VNR Vignana Jyothi Institute of Engineering and Technology Student Paper	<1 %
28	"Driver Drowsiness Detection System using Machine Learning Algorithms", International	<1 %

Journal of Recent Technology and Engineering, 2020
Publication

29	R. Feraund, O.J. Bernier, J.-E. Viallet, M. Collobert. "A fast and accurate face detector based on neural networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001 Publication	<1 %
30	Submitted to Universiti Sains Malaysia Student Paper	<1 %
31	Submitted to British University in Egypt Student Paper	<1 %
32	Submitted to Middlesex University Student Paper	<1 %
33	Submitted to Queensland University of Technology Student Paper	<1 %
34	mdpi-res.com Internet Source	<1 %
35	Submitted to University of Missouri, Kansas City Student Paper	<1 %
36	upcommons.upc.edu Internet Source	<1 %